Advancing Software Development Efficiency in an Open Source Software Context

Orucevic-Alagic, Alma

2013

Advancing Software Development Efficiency in an Open Source Software Context

Alma Oručević-Alagić

Licentiate Thesis, 2013

Department of Computer Science
Lund University

ISSN 1652-4691
Licentiate Thesis 1, 2013
LU-CS-DISS:2013-1

Department of Computer Science
Lund University
Box 118
SE-221 00 Lund
Sweden

Email: Alma.Orucevic-Alagic@cs.lth.se

Typeset using LaTeX
Printed in Sweden by Tryckeriet i E-huset, Lund, 2013

ABSTRACT

Open source software has been gaining popularity, especially among commercial organizations. The broad industry acceptance is in large part due to the demonstrated ability of open source solutions to compete with proprietary alternatives. Using open source software components enables companies to reduce their own development costs, and thus improve software development efficiency. Many open source business models have emerged around open source software, the majority of which focus on service oriented revenue models while some include parts of proprietary software, i.e., hybrid business models. In a global economy, where many companies have offices and software development resources distributed in different geographic locations, development practices used by open source community pique industry interest.

Given the dynamic of the open source software phenomena and commercial interest in the area, research focus is needed to understand the fabric of open source communities. In particular, more research is needed to understand motivation, inner workings, and different consequences of collaborative industry partnerships observed within open source communities. In addition, more evidence is needed to understand criteria and implications for selection and usage of open source software components within a commercial setting. Finally, more research is needed to define methodology which can be used to assess and measure the applicability of open source development practices within a closed development setting process.

The research effort presented in this thesis has used both the qualitative and quantitative empirical approach to study the phenomena.

The preliminary results show that open source software is a viable alternative to proprietary software. It helps companies reduce development costs by using an open source component as a type of off-the-shelf component for non-differentiating parts of software products, and thus providing an opportunity to focus development efforts on differentiating aspects of their software products. Analysis of the open source development process have shown some major differences as compared to traditional development practices. Many companies have found pragmatic ways to collaborate with open source communities, thus reduc-

ing the cost of software development and creating new business models. A new approach based on social network analysis has been proposed and applied to assess the Android committersÕ network structures and measure cross-collaboration and influences within the networks.

The future research will focus on defining quantitative approaches based on network theory to assess the structure and evolution of software development effort by studying open source projects as well as the projects implemented within a closed company setting. The overall goal is to implement the approach within a closed setting in order to test if it can better assess effectiveness of software development process by uncovering efficiency issues and offering possible solutions.

CONTENTS

PREFACE

List of Included Publications

The following publications are included in this licentiate theses:

I **A systematic review of research on open source software in commercial software product development**
Martin Höst, Alma Oručević-Alagić
Journal of Information & Software Technology, 53:6, pp. 616-624, 2011

II **Usage of Open Source in Commercial Software Product Development - Findings from a Focus Group Meeting**
Martin Höst, Alma Oručević-Alagić, and Per Runeson
International Conference on Product Focused Software Development and Process Improvement (PROFES) 2011, pp. 143-155, 2011

III **A Case Study on the Transformation from Proprietary to Open Source Software**
Alma Oručević-Alagić, Martin Höst,
Extended version of 6th International IFIP WG 2.13 Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, May 30 - June 2, 2010, Proceedings, pp. 367-372, 2010

IV **A Case Study of Open Source Development Practices within a Large Company Setting**
Alma Oručević-Alagić, Martin Höst
Submitted to a journal 2012

V **Network Analysis of a Large Scale Open Source Project**
Alma Oručević-Alagić, Martin Höst
Submitted to a journal 2013

Contribution statement

The author of this licentiate theses, Alma Oručević-Alagić, is main contributor of publications III-V. For these papers, she was the main designer, implementer, and responsible for most of the writing and the running of the research process. For the Paper II, the author participated in the design and implementation of the focus-group meeting. The author was involved in the implementation of the research presented in Paper I, the systematic literature review, in particular the articles' selection and assessment process, as well as in the writing of parts of the paper, such as summarizing two subject areas.

ACKNOWLEDGEMENTS

This work was funded by the Industrial Excellence Center EASE - Embedded Application Software Engineering.

I would like to express my utmost gratitude, to all the people without whose help this licentiate thesis would not be possible. Thank you Prof. Dr. Martin Höst, for your invaluable guidance, insight, and patience during my journey of knowledge acquirement. It has helped me look at the field of research from a different perspective and increased my awareness on the importance of objective and structured research approach. My thanks also go to Prof. Dr. Per Runeson, whose guidance and leadership by example is an inspiration for everyone who has had an opportunity to work with him. I would also like to thank all my colleagues at the department of computer science at Lund University, especially members of the SERG for the inspiring discussions and helpful advises. I am very grateful for having the opportunity to work with such a wonderful group of people. In addition, my special thanks go to faculty members with whom I had an opportunity to work with and learn from, especially Prof. Dr. Görel Hedin, Dr. Lennart Anderson, and Per Holm. I would also like to thank the members from the companies supporting EASE project, without whose help much of the research would not be possible.

Finally, I would like to thank the most those who have to put up with me on daily basis, and that is my loving family. I would especially like to thank my parents, my mother Sabira, and especially my father Ramiz, who has truly been the most inspiring person in my life. His kindness, patience, and willingness to spend a lot of time with me showing me the importance of education and honorable conduct have been the most valuable gift a child could ask for. Special thanks also go to my aunt, Fatima Druškić, who has been that very special sunshine in my life. Last, but not the least, I would like to thank my wonderful husband Amir, for his support, patience, and all the joy he brings to my life's endeavors. My girls, Aiša, Emina, and Selma, have made this life journey truly worthwhile. At the end, my deepest thanks go to the God, the Lord of the worlds, the omnipotent and the omnipresent, and the provider of all good in life.

INTRODUCTION

1 Introduction

1.1 Software Development Efficiency: Open Source Software Perspective

Improving software development efficiency has been a daunting and a complex task. Different software development methodologies and processes have been proposed and applied to pragmatically manage complex and entangled software development entities comprised of resources, requirements, software development artifacts, and schedule [PC86]. Waterfall development [Roy87], rapid prototyping [LG97], iterative development models such as spiral model[Boe88] and incremental delivery [LB03], rational unified process [Kru02], agile unified process[NB07], agile methodologies such as scrum and extreme programming [DD09] are examples of well known and utilized software development methodologies and processes. However, it seems that no single one of them systematically reduced the initial complexity problem. This goes in line with a statement of a well known computer scientist, Frederick Brooks, that there seems not to exist a silver bullet, that is a methodology or a process that would bring a significant improvement in software development efficiency [Fra+07].

In the past couple of decades open source software (OSS) solutions have become an important alternative to many in-house, proprietary software solutions [HOA11a]. Operating systems, database management solutions, web and application servers, development tools, and office software are some examples of types of software products that successfully compete with corresponding proprietary alternatives. Many business models have emerged around OSS, and many industry eco-systems have profoundly been shaped by OSS solutions, such as, e.g., Android open source project [Inc13]. While initially OSS development was reserved for enthusiasts contributing their free time to work under an open source community developing an open source product, industry interest in the software has

introduced some new models of open source communities. Hence, today we can see many successful open source communities flourishing under guidance and resource commitment from commercial organizations.

The dynamics of the OSS and industry involvement with the field have prompted research in order to better understand how communities of geographically distributed developers that rely primarily on electronic means for communication can organize work to develop large scale, enterprise quality software solutions. This more so given the inherent software development complexity. The goal of conducted research is to understand the fabric of open source community, i.e. what development methodologies and processes are present in successful open source communities and their applicability within a closed development setting. As means to achieve this goal, the following subgoals were identified:

1. Assess the usage, development, and business models of commercial organizations with respect to OSS.

2. Identify a framework of the most important OSS development process characteristics and assess its applicability within a closed development setting.

3. Explore applicability of network theory analysis in studying a structure and an evolution of OSS development communities.

Paper I presents a comprehensive literature review of previous research on the topic of OSS in commercial software development. Based on the results of the review and with the intent to further our understanding of current industry practices, a focus group meeting with relevant industry participants who use open source software was held. The conclusions of the meeting were analyzed and presented in Paper II. Paper III analyzes a process of transitioning the Ingres database solution [Ass09] from proprietary to open source focusing on the changes in the code's quality metrics. In order to assess the applicability of open source development practices within a closed development setting a study which identifies common open source software development practices and studies their applicability within a large company setting was conducted. The results of the study are presented in the Paper IV. Finally, Paper V presents a quantitative, network theory based approach, to study the structure of open source communities by identifying network centrality features, tendencies to form cliques, the most influential and the most central participants.

2 Background and Related Work

In the early days of software development, during the late 1940's, building large and expensive computers involved more effort and resources than creation of software that would execute on the machines. Hence, the code was shared freely

among the scientist. With technology advancement and production of more complex and diversified computing machines, the field of computer programming yielded new guidelines for creation of more complex software products. Among the first pioneers of the field were Edsger Dijkstra proposing layered architectures in 1968 [Dij83] and David L. Parnas who in 1972 [Par72] introduced concepts of system modularization. Thus, instead of building new solutions from the scratch, which was a common practice in 1970s, programmers started building and using reusable, tested and verified families/architectures, enabling them to create unique software products by introducing variance and specific implementations at appropriate levels. Hence, the ability to share software architectures increases software reuse and, as a result, it improves productivity and reduces cost of software development.

Sharing of the code became especially popular in academic environment. As described by Raymond [Ray01a], the Berkeley Software Distribution license also known as BSD, is a result of years long collaboration on the development of the Unix operating system by the University of California, Berkeley and AT&T labs. In the beginning of 1980s, especially during the time the personal computers gained popularity, the decades old concept of software source code sharing, was replaced with proprietary, closed source software products. As a response to the new situation, open source proponents led by the effort of Richard Stallman founded the Free Software Foundation (FSF) [FSF13]. However, the efforts were not met with a broader public acceptance, especially among the industry participants [Web04b]. According to Raymond [Ray01a], the emergence of the Linux operating system was a pivotal for the open source movement proliferation especially with industry, as it has demonstrated that a large open source community can produce complex and sophisticated software and that business models can be built around open source software. In 1998 Eric Raymond became one of the founders of the Open Source Initiative (OSI) [OSI13a], a non profit organization with aim to advocate and educate about the benefits of open source. The OSI also issues Open Source Initiative Licence trademark, which according to the organization's mission statement, has a purpose of building trust around all constituencies of an open source community carrying the trademark [OSI13b].

2.1 Open Source Software: Industry Perspective

A wide industry acceptance of OSS products such as the Linux and the Apache, and the emergence of OSS business models has created a need for systematic study of the OSS phenomenon from different perspectives. Research by Joel West [Wes07] discusses the aspect of software commoditization. He argues that while there exist some unique and new technology inventions coming from the OSS world, such as the Apache web server, the majority of broadly adopted OSS solutions are counterparts for existing proprietary solutions, such as, e.g., Linux and MySQL. Android is another example of enterprise grade open source operating system for mobile devices. Companies from the entire mobile eco-system have

joined Android Handset Alliance in an effort to help build and promote it. Sharing of development costs enables companies to decrease production costs of undifferentiating part of the product, in this case the underlying Android operating system, and focus resources which customers perceive as added value.

Besides using an OSS component as a third party component, companies can also choose to open source all or a part of their software product. For example, Computer Associates, the company ranked as one of the five largest software vendors [For13] as discussed by Raymond [Ray01b], open sourced the Ingres database management system. The software is used in some of the company's products, but over the time it has become a non-differentiating technology, and lost its leading edge over other commercial and OSS solutions, e.g., Oracle and MySQL [OAH10]. Hence, the motivation for the Ingres open sourcing is sharing of development burden with the community, but also regaining some of the lost market share. The Netscape web browser is one example where a company has lost a large part of its market share due to entrance of the Internet Explorer web browser distributed by Microsoft corporation. To remedy the situation, Netscape decided to become OSS in 1998 [Ray01a]. The Eclipse OSS project develops and maintains Eclipse, leading software development platform, and it is governed by the Eclipse Foundation whose members are largely comprised of leading industry software companies. By participating in the Eclipse OSS, companies can ensure that necessary functionality for building software products with their proprietary or open source solutions is included in the Eclipse. Hence, by contributing resources to an OSS project, company can also proliferate usage of its own products. This was also the main motivation for the IBM to open source Eclipse in 2001 [Wes07], which based on development effort was valued at 40 million dollars at the time.

Many studies have been conducted to understand the underlying fabric of the OSS communities. The Orbitan Software Survey [RAG00], conducted in 2000, studied the make up of participants that participate in OSS projects, such as RedHat Linux v 6.1, Linux Kernel sources v 2.2.14, Munitions Cryptography, and some 50% of projects available through Freshmeat. The results of the survey show that 10% of developers, or 1276 of them, contribute to 72% of the code base comprised in total of 25 million lines of code and 3149 distinct projects. A study by Crowston and Howison [CH05], analyzed the way participants communicate in projects hosted under Source Forge. Even though Source Forge at the time of the study hosted over 50000 distinct OSS projects, by eliminating projects that have less than 7 developers and less than 100 bugs in bug reporting system, they discovered that only 124 projects or just 0,002% of all hosted projects fit the criteria. The outcome of such selection criteria can point at the importance of understanding the studied OSS communities at an individual level, as there seems to exist many projects hosted by various OSS portals that are not active. The results, based on studying communication channels for 61068 bug reports, show that the majority of the projects have highly centralized decision and communication structures. Further more, the study also found that there exists a predictable relationship be-

tween the structure of the code and the organizational structure of the development team.

A question of what motivates participants to take part in development of OSS has been analyzed in studies at Kiel University [HNH03] and by Boston Consulting group [BCG02]. The study at Kiel University examined motivations of Linux kernel project participants from 28 countries. The two leading motivating factors identified in the study are participants' desire to increase their own commercial/-market value and personal satisfaction. A study by Boston Consulting Group, done on 525 Source Forge community members, showed results in line with the Kiel University study, with the highest motivating factors as personal belief in OSS, hope of increasing ones commercial value, and an opportunity to enhance skills.

2.2 Software Development Efficiency: Central Software Engineering Issue

Raymond [Ray01a] notes that the emergence of the Linux operating system has demonstrated the effectiveness of OSS development process, i.e., that large and complex enterprise grade software can be produced by very large number of self-managed developers using primarily electronic communication channels. He goes on to show that this defies Frederick Brooks' notion that adding more resources to a late project increases communication costs and complexity and thus makes the project even more late/less efficient [Bro95]. The argument for this lies in the setup of the traditional OSS communities that is very different from the traditional development communities found in commercial organizations. The pivotal difference, as Raymond claims, is the existence of a small core group of developers in OSS projects, and a large community of beta testers and occasional contributors that through close-nit and transparent communication can early and more efficiently identify bugs. Hence, the Brooks law in an OSS community only applies to the close nit group of developers, and not the entire OSS community. This argument has been called by Raymond the Linus Law and phrased as 'Given enough eyeballs, all bugs are shallow'.

Fogel [Fog05] presents a thorough insight into the fabric of OSS communities, by describing common open source development practices (OSDP) present in successful OSS projects. Some companies have shown interest in implementation of OSDP internally, within a closed development setting, in order to improve development efficiency. The OSDP also seems to be a good fit for global companies, having development teams in different location sites. A number of studies have been conducted to understand the applicability of the OSDP within a closed, commercial environment. The studies conducted in HP [MM08], Lucent [GGH06], and Nokia [LRM08a] analyzed development of software product within the company setting. Software developed in such way is called, e.g., inner-source or closed open source. The results of the studies show that the application of OSDP within

a company setting can have benefits such as improved reuse, improved quality, rapid developer redeployment, increased awareness of a developed software, and increased development speed.

Scacchi [Sca10] argues that OSS development is an interesting alternative approach to development of large systems and suggests that further research, especially using empirical examination, is needed in order to better understand OSS development practices (OSDPs).

2.3 Software Development Efficiency: Micro and Macro Aspects of Network Analysis Applicability

Social network analysis is a study approach based on the application of network theory on social networks for the purpose of studying structure and dynamics of social relationships [WF94]. A few studies have applied social network analysis to study communication archives of open source communities, as well as committer and module networks. Kevin et al.[HIC06a] study concludes that there seem to exists a predictable relationship between structure of the code and the organizational structure of the development team. Results of a study by Cleidson et al. [SFD05] show that the network structure of the source code is highly related to the way communication is organized.

The availability of OSS source code repositories, as well as the presence of OSS communities' communication archives makes it possible to study the OSS phenomenon as social networks. Thus, social network analysis can be used to understand, e.g., an underlying structure of a development organization, the existence of cliques, and to identify participants with high influence and centrality. As source code repositories and communication archives host historical data, this quantitative approach can be used to assess a project's evolution.

Understanding the structure and dynamics of an OSS community can be an important factor from both a macro and a micro perspective. The macro perspective assumes knowledge of the nature of governance and influences within the community, which can be especially relevant for companies planning to participate in an OSS community. As discussed in earlier sections, the companies' motivation can be to share cost of development burden or to proliferate its own technology. As more and more OSS stacks emerge, social network analysis can be used to study committer networks of distinct OSS communities whose software is bundled under an OSS project, such as the case with the Android OSS. The micro perspective deals with aspect of applying social network analysis to closed development communities, e.g., to measure how some organizational or development process changes impact underlying committer and module networks, or to understand who the most relevant and influential code contributors are, assess cross-team collaboration, etc. Hence, social network analysis can provide an insight that can help companies define appropriate strategies to improve software development efficiency on the macro and the micro level.

Table 1: Research questions with respect to the identified industry's OSS community roles

Role	Paper	Research Questions
Identification of the industry roles in an OSS context	Paper I	RQ1: What are the industry roles with respect to OSS?
User of OSS Components	Paper II	RQ2: What are industry practices for an OSS component selection process?
Implmenetor of an OSS business model	Paper III	RQ3: How does the open source community change an open-sourced proprietary software product in terms of static software quality metrics?
Implementor of OSDP within closed development setting	Paper IV	RQ4: How aligned are OSDPs with the traditional development practices ?
Participant in an OSS community	Paper V	RQ5: How can social network analysis theory be applied to access the structure of software development communities?

3 Research Focus

This section describes how the work included in the thesis can contribute to an increased understanding of the OSS development process and its relevance and applicability within a commercial organization. The relationship between the contributions are presented in Figure 1. In order to gain a better understanding of how OSS can be used to improve software development efficiency, a systematic review on the topic of usage of OSS was conducted. The results of the review, presented in Paper I, show that there exist four distinct roles that industry participants have taken with respect to the OSS. A commercial organization roles with respect to an OSS community are identified as:

1. User of OSS components, as a part of component based software engineering.
2. Implementor of an OSS business model.
3. Implementor of open source development practices (OSDPs) within own closed development setting.
4. Participant in an OSS community.

The roles are further examined through more specific research questions (RQs), as presented in Table1

The identified roles also represent different aspects or categories on which research could focus. In many cases the roles tend to be overlapping, e.g., a company using OSS components is often also a participant in the OSS community. Hence, focusing research exclusively on one category, ignoring the roles' synergy, can result in an inadequate and partial assessment of the OSS with respect to commercial context. Paper IV presents a case of a large, global software and hardware company that bases its product on an OSS, and as a consequence, also participates in the OSS community. This synergy has also created competence within the company on how the OSS community works and the underlying OSDP it implements. Recognizing potential benefits of the OSDP, some of the company's internal development roles and processes were modified to resemble those found in the OSS community. Hence, the studied company fills the four OSS roles, in an effort to increase its development efficiency.

Paper II further explores industry prerequisits in relation to the use of OSS components. Software components that are not maintained deteriorate according to Land [Lan02], so the virtue of successful use of OSS components assumes concerns such as component modification and bidirectional updates, from the company to the OSS community and vise-versa. Paper III studies the transitioning process of a proprietary database product, Ingres, into company sponsored OSS community focusing on how the OSS community impacts different software quality metrics. This case study also brings further clarification on the OSDP. In particular, the transitioning process shows the types of modifications that need to be made to the proprietary solution in order for it to become compliant with OSDP, such as a creation of complete and up-to-date documentation, setup of intuitive and accessible OSS project portal infrastructure, installment of appropriate resources that govern community.

The work presented in Paper V was motivated by the need of quantitative approach for the assessment of an OSS community. In particular, understanding the structure of the community, the most central and influential participants, its centrality features and tendencies to form cliques. As discussed in Section 2, there has been some work done on the applicability of social network analysis on the study of development communities. In this research we propose an approach where the networks are studied as directional and valued graphs, and argue that this approach can result in higher assessment accuracy.

4 Method

The research presented in this thesis is based on empirical research, which according to Easterbrook [Eas07] implies that the research questions are related to the class of *knowledge questions*, i.e. the questions focused on the observable and measurable state of the world. The research is of exploratory nature, which according to Eeasterbrook, is typical for the early stages of the research, when an

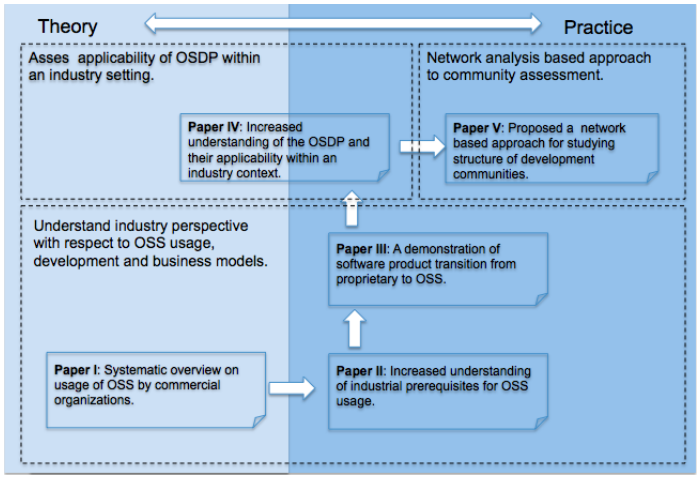

Figure 1: Overview of Contributions of Thesis

attempt is made to understand the studied phenomena. According to Kitchenham et al. [Kit+02], the exploratory studies are an important instrument for formulation of hypothesis questions and an aid in the planning of the future research activities. The exploratory research conducted in this paper is of qualitative and quantitative nature. Since software engineering involves a human factor, according to Seaman [Sea99], qualitative studies are necessary to study complex phenomena such as the ones that involve human behavior. Seaman also recommends to complement qualitative methods with quantitative methods. Qualitative and quantitative empirical software engineering can be conducted in the form of systematic reviews, a surveys, action research, experiments, or case studies.

According to Kitchenham et al. [Bre+07], systematic reviews can be used, among others, to provide a framework to appropriately position new research. A systematic review is divided into three stages; the planning, the execution, and review phase during which review protocol is created and validated, appropriate relevant research is identified, assessed for quality, and synthesized. Paper I is conducted as systematic literature review to provide background on the current state of research with respect to industry roles within the OSS world.

The research presented in Paper II was conducted in the form of a focus-group meeting, which according to Kontio [KLB04], is an effective method to obtain qualitative insights and paracticioner feedback. However, as the data obtained in such way is limited in time and scope, it is suggested that this type of research be complemented by another more rigorous methodology. The results of the Paper II, have been shown to complement the result obtained from the systematic review presented in Paper I.

Table 2: Research Type and Method Used in the Papers

Work	Research Type	Research Method
Paper I	Exploratory	Systematic Literature Review
Paper II	Exploratory	Focus-group Meeting
Paper III	Exploratory	Case Study and Quasi-Experiment
Paper IV	Exploratory	Case Study and Survey
Paper V	Exploratory	Action Research

The research Paper III is conducted as a case study with a quasi-experiment component, which according to Eeasterbrook [Eas07], is a variant of an experiment performed when the conditions for a true experiment are not feasible. Since the event of open-sourcing the Ingres solution was performed in the past, and the event could not be recreated, this quasi-experiment methodology was deemed as appropriate to use. Paper IV, is conducted as a case study with survey elements. According to Runeson and Höst [RH09], case study is appropriate methodology to use when observing a phenomenon within its natural context, and it can be combined with survey.

Finally, the paper V is conducted as action research, which according to Wieringa [Wie12], consists of developing new techniques for software engineering and evaluating them for the purpose of continues improvement. In paper V we propose a new approach to the application of social network analysis for the purpose of assessing committers networks. Hence, the action research is done with respect to finding appropriate approach to create the committers' networks and then testing the approach with Android. Table2 provides a summery of research methodology type and research method used. Figure 2 provides an overview of the thesis focus with respect to the research methodology used. More specifically, it shows links between results of earlier studies in context of them being used as research topics in later studies. Hence, e.g., the results of study presented in Paper I, were used as research topics in Paper II, Paper III, and Paper IV. Further more, findings presented in Paper III and Paper IV motivated research topics of study presented in Paper V.

5 Results

This sections presents the results of the conducted research from each of the included papers.

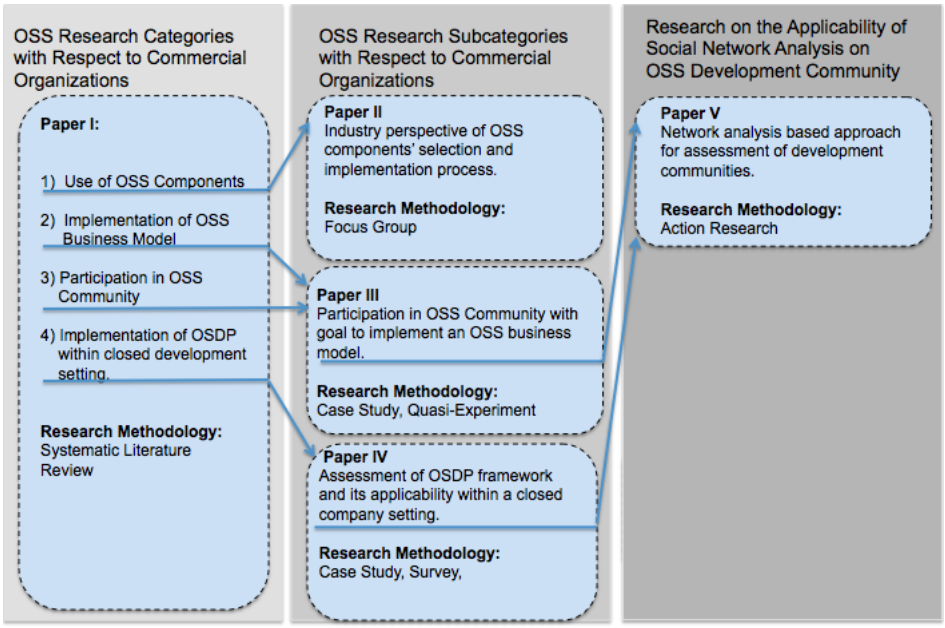

Figure 2: Research methodology and relationships between the exploratory research subcategories

5.1 Paper I: A Systematic Review of Research on Open Source Software in Commercial Software Product Development

The aim of this study was to conduct a comprehensive systematic review of research on usage of OSS components and OSDP within the commercial context as well as the industry participation in the OSS communities. We have identified and reviewed a total of 495 articles, 357 of which were found through an automated search of INSPECT and COMPENDEX databases which include articles from major conferences, journals, and publishers (e.g. IEEE, ACM, Springer, IEE). The remaining 138 reviewed publications were identified through a manual search and include all, at the time available, articles from the Conference on Open Source Systems. By applying rigorous methodology, we have identified 23 most relevant publications that can be divided into four categories: OSS as a part of component based software engineering, business models based on OSS, company participation in open source communities, and usage of OSDP within a company setting. The research methodologies used in the identified articles are equally divided between case study and survey. The results of the research by Lundell et al. [LLL06] show that 75% the companies that use the OSS components for development purposes, also participate in the OSS communities. Another research shows that 6-8% of the Linux Debian GNU code base is contributed by companies [RDGB07]. The conducted research also shows that there is not enough research done on how OSS communities function [Bon+07]. While there is evidence of successful business models built around OSS, especially for hybrid models [Wes03], there exist many challenges in sustaining development communities. According to Koening [Koe09], open source is conducive to creating competitive advantage by shifting from traditional revenue model based around a program's functionality, and focusing on service oriented revenue models. The results of the systematic review show that companies engage in an informal process when it comes to selecting an OSS component to use internally [Li+09]. Finally, studies conducted on using OSDP within a company setting can facilitate innovation Gurbani et al. [GGH05], Lindman et al. [LRM08b].

5.2 Paper II: Usage of Open Source in Commercial Software Product Development - Findings from a Focus Group Meeting

The objective of this study was to get relevant industry views on prerequisites for using OSS components within a commercial setting. For this purpose, a focus group meeting was held with industry representatives, and the meeting's discussion was based around predefined questions. The questions had two primary concerns: to understand the selection process of OSS components, and to understand how modifications of OSS components were handled. The discussion input was

collected in form of notes, that were later classified and summarized, and thus results of the research were presented in the summary form. The main identified aspects include:

1. Concerns related to the discovery process of candidate OSS components'.
2. Management of the decision making process pertaining to the selection of best fit OSS candidate with respect to legal, technical, and community support aspects.
3. Practices and issues with respect to management of OSS component modifications.
4. Advantages of participating in the OSS community.

The findings of the research are in line with findings from the literature review in Paper I. The results also confirm that when a company decides to use an OSS component, it is important to establish ties with the respective OSS community in order to acquire skills needed to maintain the component. In order to ensure compatibility with future releases of the OSS component, it is important for a company to give back to the OSS community changes made to the OSS component.

5.3 Paper III: A Case Study on the Transformation from Proprietary to Open Source Software

Paper III presents a case study on the transition process of Ingres database management system from proprietary to open source. The research provides an insight into business motivation to open source the proprietary solution, concerns and issues that need to be taken into account during the open-sourcing process, as well as an outcome of a well planned transition process. The focal point of this research is measuring change in software quality metrics, e.g., effective lines of code, cyclomatic complexity, and file function count between the proprietary and the open source community modified version of the product. The software metric changes were tracked for separate groups of the Ingres source code modules grouped by their functionality into the back end, front end, common, and utilities components. The research shows that the majority of the new code added and changed by the Ingres open source community was located in the front end module, while the smallest number of changes were made to the backend module. The software quality metrics such as cyclomatic complexity, effective lines of code were improved after the transition to open source. The open sourcing process resulted also in a 100% increase in customer base, and 32 %increase in revenues. While the investigated software presents just one case of the proprietary software product open-sourcing process, and thus can not be freely generalized, it offers a valuable insight into open-sourcing concerns, business motivation, and the effect of community on the company sponsored open source product.

5.4 Paper IV: A Case Study of Open Source Development Practices within a Large Company Setting

Based on the previous research, and motivation to explore the applicability of OSDP within an industry context, a case study was designed for this purpose. The study was conducted in a large software and hardware developing company that bases its software products on a large open source solution. Through a long involvement with the open source project, the company has modified some of its development practices to be more in line with the open source community development processes, and thus represented a good case candidate to study the applicability of the OSDP in a closed company setting. The purpose of the study was two-fold: identification of a common OSDP framework based on evidence and practice, and the assessment of the framework compatibility with the case company's development processes. The author of the thesis was granted company and network access to study the company's portal and internal documents related to past and ongoing projects, as well as development practices and guidelines. The results of the portal and documentation review were validated through semi-structured interviews conducted with the case company's employees. The research results show that many of the processes and roles characteristic to OSDP, such as common development portal and electronic communication infrastructure, formally are present in the company. However, in practice they are very little used, and the development process seems to fit more traditional development than the open source context. Furthermore, results of the research show existence of roles in line with the roles found in open source communities, but with additional and conflicting tasks, e.g., ensuring soundness of technical solution and not sacrificing it due to a time constraint and being project lead working under the time constraint. The research was conducted as one part of a two part prolonged research. The completed, first part of the research presented in this paper, is planned to serve as an input into second part, where certain modifications will be made to the company's development practices based on results. The second part of the research would then observe the outcome of applying of the changes to the overall development efficiency.

5.5 Paper V: Network Analysis of a Large Scale Open Source Project

Research conducted in paper V, was motivated by a need to define a quantifiable approach for assessing development communities in terms of source code committers' structure, influence, centrality, and cross project collaboration. The study focuses on the Android Open Source project community, an interesting case to examine, not only because of the OSS product's broad use by almost entire mobile eco-system [Goo13] and its leading market share position [GS13], but also since it is built as an OSS stack, thus including over 150 other open source projects.

This work also proposes a new approach for using social network analysis to study open source project committer networks. The new approach is the result of a study on social network analysis theory and the existing research within the field. By applying the existing analysis procedure on the data extracted from the Android source code repository, we came to the conclusion that the results do not accurately represent the studied community. The main reason for such outcome, lies in the fact that the existing social network analysis procedures, study committer networks from a perspective which does not take into account the committers' weights relative to the source code change event, i.e., it does not take into account the number of times a committer has made a change to a source file. The results of the research show that in an industry sponsored open source project, the company exhibits large control over the source code, even in the other OSS projects that are included in the stack and whose communities are not led by the company. The implications of such finding are relevant as they show one example of how a large company through an OSS project can impact industry participants, as well as assert influence in projects that are not directly under its guidance. This information is relevant for the companies who plan to undertake similar projects or join an OSS community. The proposed approach can also be applied to study structure and evolution of any software development community.

6 Synthesis

This section presents a synthesis of the results of the thesis with respect to research questions discussed in the section 3.

RQ1: What are the industry roles with respect to OSS?

Based on the systematic literature review presented in Paper I, the industry roles with respect to OSS can be divided into four distinct categories: usage of OSS components, participation in OSS communities, implementation of OSS business models, and application of OSDP. The research has shown a combination of the roles practiced concurrently. As discussed in Paper II, the companies using OSS product are motivated to participate and contribute code to an OSS community due to future update functionality and community support for the product. Some of the companies that have built competence and knowledge of the OSDP, tend to implement some aspects of the OSDP, as is discussed in the Paper III. Many large global companies have recognized business opportunities with respect to OSS, and thus there exist many examples of large companies that have built successful business models around an OSS product.

RQ2: What are industry practices for an OSS component selection process?

The industry has recognized that an OSS component can be used as an alternative to a proprietary solution or off-the shelf component. While there is a shared pool of concerns when choosing any type of a third party component, there are some specific concerns identified for the OSS component, such as community

support, legal issues, code maintainability, and handling of future updates. Results of Paper II show that the industry practices for an OSS component selection exhibit ad hoc characteristics. The selection criteria assumes gathering information through informal means, e.g., searching web forums and OSS development communities. The results of a focus-group meeting presented in the paper II show that industry has recognized the importance of building competence with an OSS community.

RQ3: How does the open source community change an open-sourced proprietary software product in terms of static software quality metrics?

Results from the case study on a transition of the Ingres database management system from proprietary to open source, as presented in Paper III, indicate that a community can have a positive impact on the static source code quality metrics. In addition, over half of the source code changes were done in the front end module, and very little code was changed in the database back-end module. While Ingres database presents only one case of open-sourcing process, it offers a relevant insight into business motivation to open source as well as points to open-sourcing concerns that need to be addressed in such process.

RQ4: How aligned are OSDP with the traditional development practices?

The research presented in paper IV shows the most important main characteristics of the OSDP found in large and active OSS communities. The OSDP assume existence of portal infrastructure which enables participants to easily use and develop software products. This portal architecture hosts an up to date information on community governance and development guidelines, as well as the software product information and documentation. The main differences observed between traditional and OSDP exist in OSDP's transparent communication process and constant feedback loop between the core developers and beta testers of the OSS product. In paper IV we present a case study of a large and global software and hardware company that bases its products on an OSS product, and which has over years built competence with the OSS community. As a consequence, the company has formally adopted some of the OSDP characteristics, but in reality this partial adoption has created some issues, such as roles with conflicting tasks. The research conducted in HP [MM08], Lucent [GGH06], and Nokia [LRM08a] shows that implementation of OSDP within a company setting can be beneficial, especially from the open innovation perspective.

RQ5: How can social network analysis theory be applied to assess the structure of software development communities?

In Paper V we show that field of social network analysis can be used to understand the underlying structure of committers network. Research by Cleidson [SFD05] and Howison et. al [HIC06a] indicate that such analysis can be used to assess organizational structure of development community. Knowledge of the structure and its most central and influential participants can be used as an input in a strategic decision making process, e.g., when deciding to participate in an OSS community, or to assess development structure of a closed source community. By

collecting the historical social network data on a community structure, we can construct the community's evolution model.

7 Threats to Validity

Identification of the threats that can potentially jeopardize a research validity is of utmost importance, especially in the field of empirical software research context where observations and measurements of the studied phenomena are conducted in natural context. The papers presented in this thesis are analyzed for validity threats based on the classification proposed by Wohlin et. al [Woh+12]. The threats to validity category are divided into four main types: *construct, internal, external, and conclusion*.

Construct validity is related to the relationship between the concepts and theories behind the experiment and what is measured and affected. Even if it is shown that the causal relationship between the two exists, we need to question weather the measurement tools are appropriate for the investigated subject of the study. In paper IV we base our research on the comparison of the common characteristics of the OSDP with the development practices of the case company. The possible construct validity threats exist in form of inappropriate identification of OSDP characteristics, also referred to as OSDP framework, and inappropriate assessment of the case company's development practices. To reduce the threats, the OSDPs characteristics were defined based on relevant works and assessment of a mature and large open source community. In addition, the author of the thesis spent two months within the company, access the company's internal electronic resources which is also known as prolonged involvement [Run+11], a practice used to improve validity of the research. The result were validated through a semi-structured survey whose results were coded, buy the second researcher, and a company employee in a senior technical position.

Internal validity is concerned with factors that may affect the dependent variables without the researcher's knowledge. The case study presented in the Paper III includes a quasi experiment, that examines the effects of the source code modifications, made by the open source community, on the static source code quality metrics. One of LehmanÕs laws [Leh80] states that a product which is not rigorously adapted or changed will, over a period of time, see decrease in software quality metrics. However, since there is no available data on the average change in software quality metrics for the studied type of the software we can not compare the observed change to some average change value. However, since the transition into open source community was a major event in terms of software maintenance, it is not probable that the transition had no affect on the software quality metrics.

External validity is related to the ability to generalize the results of the this study. The research presented in Paper II is based on the focus-group meeting whose participants were industry representatives. Hence, the results of the re-

search are based on the personal opinions, which my not be in line with a view of organization they represent. Thus, there exist a probability that these results can not be generalized or that they might not be applicable to other organizations. According to [Rob02], the extreme individual opinions tend to be offset by group reactions to them, and group dynamics can be facilitating to focus discussion on relevant issues. In the context of the thesis, the findings of the focus-group are in line with the results from the systematic review presented in Paper I, on the concerns commercial organizations have with regard to selecting an OSS component.

Conclusion validity is concerned with the possibility to draw correct conclusions regarding the relationship between treatments and the outcome of an experiment. The static software quality measures, as presented in Paper II, did not follow normal distribution, and thus in their analysis test of lower statistic power then the t-test were used. However, since the number of the analyzed data points can be considered high, the chance of detecting difference in distributions even when using non-parametric tests, is high as well.

8 Agenda for Future Research

In this section, directions for the future of the research are discussed. The two most recent papers, Paper IV and Paper V, discuss applicability of the OSDP within the industry context and propose a quantitative approach for the assessment of the development communities, respectively. The ambition is to further explore the two themes, as they are of complimentary nature. Sacchi [Sca10] defined that more empirical studies are needed to understand the applicability of the OSDP within a closed development setting. In order to be able to measure changes in the structure of a development community, a quantitative method is needed. While the community structure assessment approach presented in the Paper V provides information on some important social network metrics, it lends itself to further investigation. Planned research with respect to the results of Paper IV, and Paper V are presented in this section in form of Future Research Questions (FRQ).

9 Context and FRQs

The research approach for studying community structure, presented in Paper V can be extended. The observed social network metrics can be analyzed from the time perspective, thus creating an evolutionary view of the community. One possible exploration route is to obtain evolutionary metrics for many large and active communities, and assess distribution data from a statistical perspective in order to test the existence of statistically significant patterns or causality relations. Based on the results of paper V, a research was initiated in a large mobile company that would like to test its applicability internally. The company is planning to make

some significant organizational changes that will impact the way software is developed. The plan is to obtain committers data from source code archives, and to compare changes in developers networks for the pre-change, and the post change time. Hence, the social network metrics will need to be analyzed from the evolutionary perspective, and statistical analysis will need to be applied in order to detect significant changes. Another interesting topic is to assess cross-team work. Some FRQs that can be formulated around planned research include:

- FRQ1:Is there evidence for existence of statistically relevant patterns in committer networks' evolution data?
- FRQ2:What metrics are suitable for measuring the cross-project collaboration?
- FRQ3:Can historical data on committer networks' be used to predict the future trends in a development community?

As discussed in the section 3, Paper IV is a part of a planned two part research with the case company. Potential future research would focus around a project and its corresponding development community run in accordance to previously defined OSDP framework. The researchers would be there to assess the setup process, and to observe how the closed community works utilizing OSDP. It is probable that it will take a certain amount of time for the community participants to get acquainted with the new setup, but since the company's development resources have open source community competence, it is probable that the adjustment period will be short. The goal of the research is to observe implications, the benefits as well as the challenges, of applying OSDP within a closed commercial setting. For the purpose of the study the following FQ can be defined:

- FRQ4:What are the major challenges and benefits of OSDP implementation within a closed development setting?
- FRQ5:What are the effects of OSDP implementation within a closed setting on overall software development efficiency?
- FRQ6:What are the guidelines for implementing OSDP within a closed company setting?

The research method and approach for the identified FRQs is presented in Table 3.

10 Conclusion

In the last couple of decades, the OSS software phenomena has increasingly gained support from the commercial organizations from its potential to increase software

Table 3: Planned FRQs and Corresponding Research Method

FRQ	Description	Research Method	Type of Research
1	Detection of patterns in committers networks	Multi-Case Study	Exploratory
2	Assessment of the cross-project collaboration	Multi-Case Study	Exploratory
3	Prediction of trends	Multi-Case Study	Descriptive
4	Challenges and Benefits of implementing OSDP	Case-Study	Exploratory
5	Software development efficiency and OSDP	Case-Study	Exploratory
6	Guidelines for OSDP implementation	Design science	Descriptive

development efficiency and facilitate creation of new business models. In the systematic literature review presented in Paper I we identify four distinct roles industry takes with respect to OSS community (RQ1). The commercial organizations include OSS components in their products or use of the products for internal purposes as presented in Paper II. The usage tends to creates a relationship between the company and the community for the reasons of product support and maintenance (RQ2). By participating in OSS communities, a company can develop expertise on OSDP (RQ4) and may chose to implement some of the practices as presented in Paper IV. There exists evidence of successful transitions of OSS products from proprietary to open source community, and successful business models built around such communities (RQ3).

Ability to understand the structure and the evolution of an open source community can be an important factor that should be considered when a commercial organization plans to work with an OSS community (RQ5). Paper V proposes a new network theory based approach to study development communities and applies the approach to the study of the Android OSS project. While there has been some research done on the applicability of social network analysis in context of an OSS community structure, more work is needed to understand how the data from such analysis can be used to predict the future behavior of development communities.

As there exists little evidence on the applicability, benefits, and challenges of the OSDPs implementation within a closed development setting, we intend to conduct more studies on the subject. In particular, we plan to conduct more case studies in order to asses impact of the OSDPs on software development efficiency and propose some general guidelines that can be used for the implementation. In order to asses software development effort in context of committers' networks, we plan to further evaluate the approach proposed in Paper V, and apply it to the

closed development setting.

References

[Ass09] Matt Assay. *February 2009 Web Server Survey*. http://news.cnet.com/8301-13505_3-10156188-16.html. 2009.

[BCG02] BCG. *The Boston Consulting Group Hacker Survey, accessed on April 22, 2013*. http://mirror.linux.org.au/linux.conf.au/2003/papers/Hemos/Hemos.pdf. 2002.

[Boe88] Barry W. Boehm. "A Spiral Model of Software Development and Enhancement". In: *IEEE Computer* 21.5 (1988), pp. 61–72.

[Bon+07] A. Bonaccorsi et al. "Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research". In: *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*. 2007.

[Bre+07] Pearl Brereton et al. "Lessons from applying the systematic literature review process within the software engineering domain". In: *Journal of Systems and Software* 80.4 (2007), pp. 571–583.

[Bro95] Frederick P. Brooks. *The mythical man-month - essays on software engineering (2. ed.)* Addison-Wesley, 1995, pp. I–XIII, 1–322.

[CH05] Kevin Crowston and James Howison. "The social structure of free and open source software development". In: *First Monday* 10.2 (2005).

[Dij83] Edsger W. Dijkstra. "The structure of the multiprogramming system". In: *Communications of the ACM* 26.1 (Jan. 1983), pp. 49–52.

[DD09] Tore Dybå and Torgeir Dingsøyr. "What Do We Know about Agile Software Development?" In: *IEEE Software* 26.5 (2009), pp. 6–9.

[Eas07] Steve M. Easterbrook. "Empirical research methods for software engineering". In: *IEEE/ACM International Conference on Automated Software Engineering*. 2007, p. 574.

[FSF13] FSF. *Free Software Foundation*. http://www.fsf.org/about/staff-and-board/. 2013.

[Fog05] Karl Fogel. *Producing open source software - how to run a successful free software project*. O'Reilly, 2005, pp. I–XX, 1–279.

[For13] ForbesL. *Forbes Leading 2000 Global Companies, Sorted by Industry*. http://www.forbes.com/global2000/list/. 2013.

[Fra+07] Steven Fraser et al. “"No silver bullet" reloaded: retrospective on "essence and accidents of software engineering"”. In: *Object-Oriented Programming, Systems, Languages and Applications, OOPSLA Companion*. 2007, pp. 1026–1030.

[GS13] Market Analysis Gartner and Statistics. *Market Share: Mobile Devices, Worldwide, 1Q12*. `http://www.gartner.com/newsroom/id/2017015`. 2013.

[Goo13] Google. *Android Open Handset Alliance Members*. `http://www.openhandsetalliance.com/oha_members.html`. 2013.

[GGH05] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. “A case study of open source tools and practices in a commercial setting”. In: *ACM SIGSOFT Software Engineering Notes* 30.4 (2005), pp. 1–6.

[GGH06] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. “A case study of a corporate open source development model”. In: *ICSE*. 2006.

[HNH03] Guido Hertel, Sven Niedner, and Stefanie Herrmann. “Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel”. In: *Research Policy* 32 (2003), pp. 1159 –1177.

[HOA11a] Martin Höst and Alma Oručević-Alagić. “A systematic review of research on open source software in commercial software product development.” In: 2011, pp. 616–624.

[HIC06a] James Howison, Keisuke Inoue, and Kevin Crowston. “Social dynamics of free and open source team communications”. In: 2006, pp. 319–330.

[Inc13] Google Inc. *Android Open Source Software Project*. `http://www.android.com/`. 2013.

[Kit+02] Barbara A. Kitchenham et al. “Preliminary Guidelines for Empirical Research in Software Engineering”. In: *IEEE Transactions on Software Engineering* 28 (8 2002), pp. 721–734.

[Koe09] John Koenig. *Seven Open Source Business Strategies for Competitive Advantage*. `http://www.cs.up.ac.za/cs/aboake/sws780/references/designapproaches/collaborative/Koenig-SevenOpenSourceStrategies.pdf`. 2009.

[KLB04] Jyrki Kontio, Laura Lehtola, and Johanna Bragge. “Using the Focus Group Method in Software Engineering: Obtaining Practitioner and User Experiences”. In: 2004, pp. 271–280.

[Kru02] Philippe Kruchten. “Tutorial: introduction to the rational unified process®”. In: *ICSE*. 2002, p. 703.

[Lan02] Rikard Land. “Software Deterioration And Maintainability–A Model Proposal”. In: *Proceedings of Second Conference on Software Engineering Research and Practise in Sweden (SERPS)*. Citeseer. 2002.

[LB03] Craig Larman and Victor R. Basili. “Iterative and Incremental Development: A Brief History”. In: *IEEE Computer* 36.6 (2003), pp. 47–56.

[Leh80] Meir M. Lehman. “On understanding laws, evolution, and conservation in the large-program life cycle”. In: *Journal of Systems and Software* 1 (1980), pp. 213–221.

[Li+09] Jingyue Li et al. “Development with Off-the-Shelf Components: 10 Facts”. In: *IEEE Software* 26.2 (2009), pp. 80–87.

[LRM08a] Juho Lindman, Matti Rossi, and Pentti Marttiin. “Applying Open Source Development Practices Inside a Company”. In: *OSS*. 2008, pp. 381–387.

[LRM08b] Juho Lindman, Matti Rossi, and Pentti Marttiin. “Applying Open Source Development Practices Inside a Company”. In: 2008, pp. 381–387.

[LLL06] Björn Lundell, Brian Lings, and Edvin Lindqvist. “Perceptions and Uptake of Open Source in Swedish Organizations”. In: *nternational Conference on Open Source Systems, OSS*. 2006, pp. 155–163.

[LG97] Luqi and Joseph A. Goguen. “Formal Methods: Promises And Problems”. In: *IEEE Software* 14.1 (1997), pp. 73–85.

[MM08] Catharina Melian and Magnus Mähring. “Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard”. In: *OSS*. 2008, pp. 93–104.

[NB07] Sridhar Nerur and VenuGopal Balijepally. “Theoretical reflections on agile development methodologies”. In: *Commun. ACM* 50.3 (Mar. 2007), pp. 79–83.

[OSI13a] OSI. *Open Source Initiative Board*. `http://opensource.org/board`. 2013.

[OSI13b] OSI. *Open Source Initiative Mission Statement*. `http://opensource.org/about`. 2013.

[OAH10] Alma Oručević-Alagić and Martin Höst. “A Case Study on the Transformation from Proprietary to Open Source Software”. In: *OSS*. 2010, pp. 367–372.

[Par72] David Lorge Parnas. “On the Criteria To Be Used in Decomposing Systems into Modules”. In: *Communications of the ACM* 15.12 (1972), pp. 1053–1058.

[PC86] David Lorge Parnas and Paul C. Clements. “Correction to "A Rational Design Process: How and Why to Fake It"”. In: *IEEE Transactions on Software Engineering* 12.8 (1986), p. 874.

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O’Reilly Media, Inc., 2001.

[Ray01b] Eric S. Raymond. *The Cathedral and the Bazaar*. O’Reilly Media, 2001, p. 186.

[RAG00] Vipul Ved Prakash Rishab Aiyer Ghosh. “The Orbiten Free Software Survey”. In: *First Monday, Peer Reviewed Journal on the Internet* 5.7 (2000).

[RDGB07] Gregorio Robles, Santiago Dueñas, and Jesús M. González-Barahona. “Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time”. In: *International Conference on Open Source Systems*. 2007, pp. 121–132.

[Rob02] Colin Robson. *Real World Reserach*. 2:nd. Blackwell Publishing, 2002.

[Roy87] W. W. Royce. “Managing the Development of Large Software Systems: Concepts and Techniques”. In: 1987, pp. 328–339.

[RH09] Per Runeson and Martin Höst. “Guidelines for conducting and reporting case study research in software engineering”. In: *Empirical Software Engineering* 14 (2 2009), pp. 131–164.

[Run+11] Per Runeson et al. *Case Study Research in Software Engineering*. Wiley, 2011.

[Sca10] Walt Scacchi. “The future of research in free/open source software development”. In: *Future of Software Engineering Research*. 2010, pp. 315–320.

[Sea99] Carolyn B. Seaman. “Qualitative Methods in Empirical Studies of Software Engineering”. In: *IEEE Transactions on Software Engineering* 25.4 (1999), pp. 557–572.

[SFD05] Cleidson R. B. de Souza, Jon Froehlich, and Paul Dourish. “Seeking the source: software source code as a social and technical artifact”. In: *GROUP*. 2005, pp. 197–206.

[WF94] Stanley Wasserman and Katherine Faust. *Social Network Analysis. Methods and Applications*. Cambridge University Press, 1994.

[Web04b] Steven Weber. *The Success of Open Source*. Harvard University Press, 2004.

[Wes03] Joel West. “How open is open enough?: Melding proprietary and open source platform strategies”. In: *Research Policy* 32.7 (2003), pp. 1259 –1285.

[Wes07] Joel West. “Value Capture and Value Networks in Open Source Vendor Strategies”. In: *Hawaii International Conference on System Sciences*. 2007, p. 176.

[Wie12] Roel Wieringa. “Designing Technical Action Research and Generalizing from Real-World Cases”. In: *International Conference on Advanced Information Systems Engineering, CAISE*. 2012, pp. 697–698.

[Woh+12] Claes Wohlin et al. *Experimentation in Software Engineering*. Springer, 2012, pp. I–XXIII, 1–236.

INCLUDED PAPERS

PAPER I

A SYSTEMATIC REVIEW OF RESEARCH ON OPEN SOURCE SOFTWARE IN COMMERCIAL SOFTWARE PRODUCT DEVELOPMENT

Abstract

Context: The popularity of the open source software development in the last decade, has brought about an increased interest from the industry on how to use open source components, participate in the open source community, build business models around this type of software development, and learn more about open source development methodologies. There is a need to understand the results of research in this area

Objective: Since there is a need to understand conducted research, the aim of this study is to summarize the findings of research that has ben carried out on usage of open source components and development methodologies by the industry, as well as companies' participation in the open source community.

Method: Systematic review through searches in library databases and manual identification of articles from the open source conference. The search was first carried out in May 2009 and then once again in May 2010.

Martin Höst, Alma Oručević-Alagić
Information & Software Technology, 53:6, 616-624, 2011.

Results: In 2009, 237 articles were first found, from which 19 were selected based on content and quality, and in 2010, 76 new articles were found from which 4 were selected. 23 articles were identified in total.
Conclusions: The articles could be divided into four categories: open source as part of component based software engineering, business models with open source in commercial organization, company participation in open source development communities, and usage of open source processes within a company.

1 Introduction

Traditional software development is often perceived as a proprietary, in-house software development, with developers working in a geographically centralized or distributed company's location. Open source software is developed free of charge through a community driven development process, and as such, it is also provided to public at no cost, but under certain usage and distribution conditions. Many of the traditional software companies have tried to take advantage of the free software, not just by using the software, but also by creating business models and strategies around the open source software.

For example, in the mobile industry there are several attempts to form open source communities for development of software, such as the Android project[1] and the Symbian project[2]. Using and relying on open source software can be seen as an alternative way to reduce development costs and stay competitive. Hence, in a way, it can be compared to other similar business methods and strategies, such as outsourcing or acquirement of off the shelf components.

This open source business ecosystem, which has been growing over the past two decades, is quite complex and there exists a need to better understand many of its aspects. Some of the aspects are interesting in at least two different ways. Firstly, an organization can include open source components in its proprietary software product. This is comparable to including any other third party component, although the difference is that the component is now obtained from an open source community instead from a commercial organization. Secondly, an organization can provide its own proprietary software to open source community and that way reduce development costs in long run, reposition itself on the market, create a new source of income through new services, etc.

Already in 2001, Lerner and Tirole [LT01] identified "opening proprietary code" as an important research area, and observed that large open source projects often start based on software provided by "academic or semi-academic institutions". This motivates systematically investigating what research has been published in the area.

[1] `http://www.android.com/`
[2] `http://www.symbian.org/`

The outline of this paper is as follows. In Section 2 background on open source software and some related work is presented. In Section 3 the methodology with respect to search strategy and inclusion and exclusion criteria are presented, and the resulting set of articles is presented in Section 4. Finally, there is a discussion in Section 5, and conclusions presented in Section 6.

2 Background and related work

2.1 Open Source Software

Open source software has been around since the very beginning of electronic computing. In the early days of information technology it was quite natural and financially sound for developers to share source code among very few and very expensive computing machines. As the machines became smaller, more diversified, and cheaper, the number of developers grew, and the source code, in general, became more complex. Development of free software was especially flourishing in the academic environments. Barkley Software Distribution (BSD) is a license developed for distribution of the BSD version of the Unix operating system developed by the University of California, Berkeley, from 1977 to 1995 in collaboration with AT&T labs, as described in [Ray01c]. At the beginning of the development, code was shared between AT&T and Berkeley. Due to anti-monopoly laws at the time, AT&T could not sell software, but as the company was using it to sell phone-related services, it had vested interest in improving the software. During the beginning of the 1980:s and the market deregulation, AT&T was granted the right to sell software. In order to continue distribution of BSD Unix, a lot of code that was not developed by University of California Berkeley had to be backed off and rewritten.

Since the beginning of 1980s, the idea of close-sourced/proprietary software became mainstream, taking the place that free software sharing has held for a long time. The open source supporters went to found their own organizations such as free software foundation (FSF) founded by Richard Stallman, as described in [Web04a]. The FSF did not have desired impact on bringing back open source software development to the mainstream. However, this situation was about to change with the successful release of the Linux kernel. The system was initially developed by Linus Torvalds as part of an academic project, and with the support of the developer community it became a very complex, sophisticated software that was free for everyone to use. Eric Raymond was very much inspired by this set of events, and in his now famous book ”The Cathedral and the Bazaar” [Ray01c] he talks about the importance of Linux, as it was the very first time the open source developer community showed that not only complex and sophisticated software can be built in such way, but also that business models can be built around such way of software development and distribution.

In 1998, Raymond was one of the main contributors to the Open Source Initiative (OSI), an organization that is envisioned as open source educational and advocacy organization. Many companies have followed the suit, and decided to open source a piece of their proprietary software as a part of business strategy to deal with the competition. Thus, among the initial suitors we can find Netscape corporation, who by open sourcing Netscape internet browser tried to compete against closed source and free distribution of Microsoft's Internet Explorer [Ray01c].

In the past ten years, many companies have entered the open source business arena, using some of the business models proposed in [Ray01c]. Unfamiliar with the environment, companies had very quickly to readjust their way of doing business in order to ripe some perceived benefits of open source trends. Besides open sourcing software, companies tend to participate and contribute to open source projects, as well as adopt some of software development methodologies such distributed and voluntary based development community as open source utilizes.

2.2 Related Work

Stol and Ali Babar [SAB09] have made a review of the broad area of "open source" from the conference on Open Source Systems, OSS. They manually selected empirical papers from the conference and investigated them. The scope of the review that we present in this paper is more narrow (open source in commercial organizations) but we searched a broader set of articles (we searched articles in library databases with a search string, and we searched articles from the OSS conference manually, as explained in more detail below).

Stol and Ali Babar [SAB10] have also compiled a list of challenges in using open source as components in product development, based on a literature review. In this review, where they did not require any empirical grounding of the findings, they identified 21 challenges.

In [HOA10a] earlier results of this work is presented, based on a search in biblographic databases that was carried out 18 May 2009. The search that is the basis for this paper was conducted 14 May 2010, and it resulted in 4 additional articles.

3 Review Method

This research is carried out as a systematic literature review, based on the guidelines presented in [KC07].

3.1 Research Questions

The objective of this research is to understand the result of the research that has been carried out on the usage of open source software and open source software

development in proprietary software development organizations. Before the review, this was broken down to the following research questions:

1. What approaches and processes are applied by commercial organizations to introduce open source products in their proprietary products?
2. What approaches and processes are applied by commercial organizations to provide their software products to the open source community?
3. What experience is available from identified approaches and processes, for example, with respect to quality of the software products, cost of development for the providing organization, time taken to introduce new functionality, etc.?
4. What are the main motivations and business incentives for the procedures and processes identified in question 1 and question 2?

That is, we address the need to understand both what research that has been done in the area, what methods and approaches that exist, and what experience is available for the different methods. It should be noted that the objective of the research has not been to derive quantitative knowledge of which methods perform the best. The objective is more to understand which methods are used and how well the methods work in a qualitative way. If the field was more mature, and it could be expected to find a large number of empirical studies investigating the performance of alternative methods, it would of course be interesting to synthesize this knowledge. However, it is not realistic to find this many studies of this type. The objective of this work is instead to review the research that has been conducted, and in particular what kind of experience that is available for these kind of questions. That is, the research has elements of a mapping study (see for example [KC07]). However, since the objective is to summarize the findings and to understand the total result of the research that has been conducted, and the focus is not merely on identifying published research we classify this as a systematic review.

Open source software in commercial organization is related to a number of research questions that to some extent are relevant to the review, but where it was necessary to decide whether to include them in the study or not. One aspect that is often mentioned concerning open source is the importance of ”legal aspects”, such as licensing, intellectual property, etc. For example, there is a large number of licenses, all complying with the definition of open source described in [FF02], and the implications of choosing different licenses could be an important research field. This is an important and interesting field, which can affect both the adoption of open source practices and open source software components. However, for this study it was seen as out of scope for two main reasons. Even if software engineering is a multi-disciplinary field which includes legal aspects, we thought that it is of another kind than more traditional software engineering topics. To some extent the research questions that have to do with legal aspects are not the

same as traditional research questions. If legal aspects were included, then there are other areas that also would be reasonable to include, such as marketing and sales. Second, it would probably require extensive cooperation with researchers in legal aspects to make sure that the correct search terms were used, and that the right publication fora were searched. These aspects in combination mean that legal aspects were not included in the study.

An area where there are a number of research results available is on comparisons of usage of open source software, such as Open Office, and similar proprietary software systems. This was not seen as highly related to the research questions in this study and therefore excluded. It is, of course, interesting understand the differences, but it was not seen as relevant enough to the question of transforming developed software to open source or to the inclusion of open source software in developed software. Neither are studies on adoption of open source programs, for example as presented by Goode [Goo05], included. That is, this study is more on development of software than on the usage of existing software. In the same way it was decided not to include research results on usage of open source tools, such as Eclipse, in software development.

Another area that is not included, but still interesting and of potential interest to commercial organizations, concerns how open source practices can be transferred to hardware development. Only a few articles on this topic exist [AB09]. This area was not included since it is not mainly concerning software development.

3.2 Search methodology

Two main sources were searched for relevant articles: a broad search in academic databases; and a manual search through all articles of the Conference on Open Source Systems.

Searched academic databases

The INSPEC and the COMPENDEX databases were searched. Both of these databases intend to provide complete coverage of the area, and include articles from all major conferences, journals, and publishers (e.g. IEEE, ACM, Springer, and IEE). We believe that these two databases give a good coverage of articles in ”computer science” and ”electrical engineering and electronics”, which includes typical questions in software engineering, at least in more well known journals and conferences. However, the coverage of more business-related articles and articles on legal aspects is, as described above, more uncertain. Both databases were accessed through Engineering Village[3].

The following search string was used:

[3] `http://www.engineeringvillage2.org`

```
(({open?source} wn ALL) OR
 (opensource wn ALL) OR
 (libre wn ALL) OR
 (OSS wn ALL) OR
 (FLOSS wn ALL))
AND
((proprietary wn ALL) OR
 (commercial wn ALL) OR
 ({non?open?source} wn ALL) OR
 ({non?opensource} wn ALL))
AND
((empirical* wn ALL) OR
 (experiment* wn ALL) OR
 ({case?study} wn ALL) OR
 (survey wn ALL))
```

The search string contains three main parts, separated by `AND`-clauses. The first part states that the article must include the term ”open source” or some other synonym term that is often used, such as ”OSS”. The second part states that the article must include terms about commercial software development. The third part makes sure that the article is empirical, by searching for terms like ”empirical” and ”experiment”. The intention of the ”`*`” after `experiment` is to include also search terms as ”experimental” and ”experimentation”. According to [DGJ07] this should be sufficient in order to find most relevant articles in this respect.

A few more terms and details in the search string may have to be explained. The `?`-sign denotes any character, which, for example, means that both articles with the term ”case study” and the term ”case-study” are found. The term `wn` means that the phrase left of it should be found in the entity to the right of it, in this case `ALL`, which means all fields of database entries, such as title, abstract and key words. It would have been possible to list other entities such as abstract and title, but in this case `ALL` was chosen. Text within `{}`-parentheses are searched as phrases and a search is not case sensitive.

Manual identification of relevant articles

In addition to the database searches, all articles in all OSS-conferences (International Conference on Open Source Systems[4]) were inspected. The conference has been held annually since 2005 and all articles are available in full text (2005 online and from 2006 onwards from library databases). The selection was based on the formulated research questions in Section 3.1, which thereby means that articles matching the same kind of content as the identified with search string presented in Section 3.2 were found, but there was no check that the articles matched exactly.

[4]For more information see `http://www.ifipwg213.org/`

3.3 Selection of relevant articles

Articles are selected in a number of steps. First, all articles identified from the databases with the search string were listed with title and abstract. Since the articles have been selected with a search string in a database there are many articles that are not relevant. Therefore, articles that are not relevant, based on our interpretation of title and abstract, were removed. That is the articles that were not relevant compared to the research questions wee removed.

After this, the remaining articles were downloaded and read in full text. Based on this, more articles were seen as non-relevant according to the same criteria as for the title and abstract, and therefore removed.

After analysis of articles selected with the search string, articles from the OSS conference were selected manually. Since this selection was carried out after the analysis of articles identified with the search string, it was possible to use knowledge that was gained from analysis of articles identified with the search string.

All articles that so far have been selected were analyzed with respect to research methodology implementation and presentation. Three different classes were used for this:

Class A: In this type of article the research is presented in a way that makes it very likely that it was conducted according to normal requirements on empirical research methods in software engineering.

Compared to the quality assessment criteria presented used by [DD08] and [CAC08] the answer is positive to most evaluation questions, especially concerning whether it is research or merely experience report, if there is a clear statement of aims, if there is an adequate context description, if the research design is appropriate for the questions, if the data collection was appropriate for the questions, if the data was analyzed with sufficient rigor, and if there is a clear statement of findings.

This class includes both articles that do reference empirical software engineering research method descriptions, such as [RH09], and articles that do not explicitly reference this kind of descriptions.

Class B: This class of articles may not be presented as a typical article on empirical software engineering even if the overall impression of it is that it was carried out in this way. That is, all aspects of a typical article of class A may not be included, but the main impression of the paper is that the research was carried out according to normal requirements on empirical software engineering.

Class C: For this type of article our interpretation is that the researchers have not followed any traditional research method during the research. The reason may be that the presentation forum is not suitable for presentation of structured research methods or there may be other reasons.

These steps are further presented in Section 4 and illustrated in Figure 1.

3.4 Data Extraction and Synthesis

Articles of class A and class B were treated equally, while articles of class C were not further included in the review. Data from the identified articles were derived by defining categories of articles and summarizing the research in each category. Both authors first defined categories individually and then a final set of categories was defined based on discussion between the authors. The summaries were developed first by one author and then updated based on discussion between the authors.

3.5 Phases

The search was conducted in two phases. First in phase 1 the databases were searched in 2009, and the result was summarized and presented in [HOA10a]. Then the same search string was used again in phase 2 in 2010, and the results were updated with the new articles that were identified.

4 Results

In this section the actual results of the steps presented in Section 3 are presented. The results are summarized in Figure 1 and below. The research was conducted in two major phases as described in Section 3.5.

4.1 Phase 1

First the academic databases were searched on 18 May 2009, which resulted in 357 articles. However, among these articles there were a number of duplicates because the searches were made in different databases. Duplicates were identified with a simple java-program based on titles. After this, 237 articles remained (i.e. the result of step 1 in Figure 1).

After this, unrelated articles were removed based on both title and abstract. First an attempt was made to remove articles based only on title and then based on abstract, but it was not seen as possible to remove an article only based on the title. Therefore both titles and abstracts were studied during this process. After this, 45 titles remained.

After this, one additional duplicate was identified where the title was written slightly differently in different databases (”&” instead of ”and”), which means that 44 titles remained (i.e. the result of step 2 in Figure 1).

The first author of this paper first conducted these steps, and then the second author reviewed the result. None of the previously excluded articles were reintroduced.

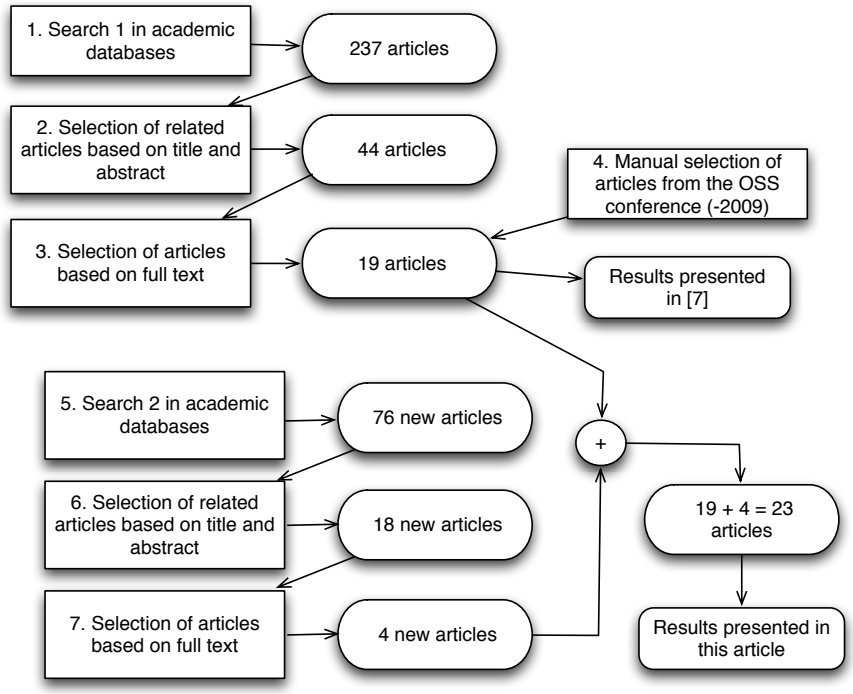

Figure 1: Summary of article selection process.

In these steps, some articles were found for which it was hard to decide whether to keep them or not based on the title and abstract. In these cases we decided to keep them to the next step instead of removing them. That is, articles that were hard to judge based on only title and abstract were kept to the next step, where the whole articles were read.

After this, the identified articles were obtained from the library database, and they were reviewed in full text. Here, some articles were removed since they were not really related to the research questions or because they were not available in full text from the databases.

Some of the removed articles were about developing open source in general, which was not seen as relevant for this work. Some were about using open source software in general, without seeing the context as an IT system that is built.

To this list of articles, relevant articles from the OSS conference were added. There was no overlap between these manually found articles and the articles that were found through the search in the databases. After this, a final set of 19 articles remained (i.e. the result of step 3 in Figure 1).

In the analysis of the papers it was clear that both anticipated and unanticipated areas were covered by the identified articles. That is, some papers dealt with questions that we thought of before, and therefore were aware of when the research questions were formulated. Other areas were more unexpected, mainly the articles about transferring the open source development process to the internal work in a non-open source product. Articles of both types were of course included in the study as long as they were seen as relevant compared to the formulated research questions.

One paper for which it was hard to judge the relevance for this study is the paper by [Kri06], which concerns motivation of developers, to some extent discussing both unpaid and paid developers. Even if the question of motivation for open source developers in general is out of scope of the review, the discussion about paid and unpaid developers makes it more relevant. However, we decided not to include the paper since it was seen as a too small part of the article. Also, concerning the paper by [Lea+02] it could be argued that this type of article should be included. The main focus of it is on design of a modular system for data analysis, but they also conclude that working with the system as OSS improves the possibility of collaborating between different universities, government agencies, and private industry, both nationally and internationally. However, this was stated as a minor part of the paper, which means that the article was not included.

4.2 Phase 2

The databases were searched with the same search string as in phase 1 once again 14 May 2010. This resulted in 76 new articles that were not identified in phase 1 (i.e. the result of step 5 in Figure 1).

The title and abstract of the articles were studied in order to remove articles that were not of interest. This was done as a cooperation between the two authors. After this step, 18 of the new articles were kept (i.e. the result of step 6 in Figure 1).

The next step was to download all articles and study them in full text in order to decided whether they really are of interest with respect to the research questions, and of type A or type B. Some articles could not be downloaded from the library databases, but most could. After this step 4 of the new articles remained.

4.3 Analysis of identified articles

After phase 2 there were in total 23 articles, i.e. 19 from phase 1 and 4 additional from phase 2.

In the rest of this paper, the selected articles are referred to with the keys that are presented in bold in appendix. For example, the first identified article is referred to as [Arhippainen03].

Table 1: Identified articles

Article	Research methodology	Class	Identification phase
[Arhippainen03]	case study	A	1
[Ayala09]	survey	A	1
[Bonaccorsi05]	survey	A	1
[Bonaccorsi06]	survey	A	1
[Bonaccorsi07]	survey	A	1
[Gaughan09]	case study	B	2
[Gurbani06]	case study	A	1
[Harison10]	survey	A	2
[Henkel08]	interviews and survey	A	2
[Hauge07]	survey	A	1
[Hauge09]	case study	A	1
[Li05a]	survey	A	1
[Li05b]	survey	A	1
[Li06a]	survey	A	1
[Li06b]	survey	A	1
[Li09]	summary	A	1
[Lindman08]	case study	A	1
[Lindman09]	case study	A	1
[Lundell06]	survey	A	1
[Munga09]	case study	B	2
[Robles07]	case study	A	1
[West03]	case study	B	1
[Westenholz06]	case study	A	1

The research methodologies that were used in the identified articles are summarized in Table 1. This is our interpretation of the chosen methodology after reading the articles. In some cases it was very clear which methodology that was used, but in other cases it was somewhat harder. For example when a set of interviews was conducted in different organizations we classified this as a survey, since we wanted to classify according to commonly used methods. However, it could had also been classified as something like ”interview study”.

In Table 1 we also present our interpretation of the classification of the method implementation (A or B). Our experience is that it is hard to evaluate articles in this way. Since no difference has been made between the two classes in the analysis, this classification should only be seen as our interpretation of the article for the purpose of this review.

The article [Li09] requires some further explanation. Since it is a summary of

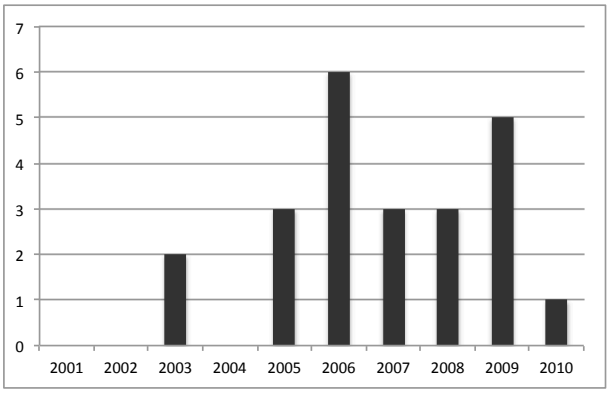

Figure 2: Publication year for included articles

the other articles by the author ([Li05a]–[Li06b]), the methodology is presented as ”summary”, and even if it is presented as a popular science article we have classified it as class A and thereby included it in the study based on the contents of the other articles.

In Figure 2 it can be seen that the oldest identified article is from year 2003, and that the most recent is from 2010. Here it should be noted that the last search in the database was conducted in may 2010, which means that there may be more articles published in 2010. It can be noted that all the identified articles are published rather recently (i.e. since 2003). There were rather many articles published in 2005 and 2006. Four of these were by the same author on the same subject.

4.4 Investigated research areas

Introduction

The articles can be divided into a number of main areas based on the contents of the articles. The formulation of content areas was done without the explicit objective to adhere to the identified research questions defined in Section 3.

The categories were defined based on the contents of the articles. That is the whole articles were used and no specific common parts of the articles were derived with a data extraction template. Each article was sorted into the category where it belonged the most even if it could be argued that some articles to some extent were related to more than one category. However, there was no article where it was really hard to decide the category or where we thought that it was equally related to more than one category. This is probably natural since the categories were defined based on the contents of the articles, and the objective during this process was to define categories based on the articles. It should be noted that the classification

was first conducted in phase 1 based on the 19 articles that were identified in that phase. It was rather easy to classify the 4 new articles in phase 2 in the same way, which means that the same classification scheme (i.e. the four areas) was kept in phase 2.

The identified categories are listed below. For each category the articles related to it are listed.

- *Company participation in open source development communities:* [Bonaccorsi07], [Hauge07], [Henkel08], [Lundell06], [Robles07].
- *Business models with open source in commercial organizations:* [Bonaccorsi05], [Bonaccorsi06], [Harison10], [Hauge09], [Lindman09], [Munga09], [Westenholz06], [West03].
- *Open source as part of component based software engineering:* [Arhippainen03], [Ayala09], [Li05a], [Li05b], [Li06a], [Li06b], [Li09].
- *Using the open source process within a company:* [Gaughan09], [Gurbani05], [Lindman08].

The research conducted in each area is shortly summarized below.

Company participation in open source development communities

It is clear that there is company participation in many open source projects. For example [Bonaccorsi07] found that in one third of the most active projects on SourceForge there was some form of company participation. Companies can participate as project coordinator, collaborator in code development, and by providing code etc. In [Hauge07] one additional role, which is more concerned with integration of open source components, is identified.

Concerning the number of companies that participate in this kind of development, [Lundell06] suggests that a significant number of the companies marginally participate in open source community. However, the participation has increased especially in SME, compared to earlier conducted studies. Of the companies that use open source projects, 75% can be said to have ”symbiotic relationship” with the OS community. This can be compared to the investigation presented by [Rables07] that show that 6-7% of the code in Linux Debian GNU distribution over the period 1998-2004 has been contributed by corporations. That is, it is clear that a rather large part of the open source code has been provided by commercial organizations, and that those commercial organizations play crucial roles in open source projects. This is especially clear in the larger and more active projects.

It is also clear that if software should be provided to a community it is important to provide enough documentation and information to get the community members going (e.g. [Hauge07]).

One risk that could be seen by companies is that people working in the organization would reveal too much information to the outside of the organization if they work with an open source community. However, the revealing behavior of this kind of software engineers was investigated in [Henkel08] and it was found that even if the engineers identified with the community they were significantly less identified and ideological about open source than the control group of non-commercial developers. The conclusion from that research is that there is indicators of commercially harmful behavior in this kind of development.

[Bonaccorsi07] presents a list of important questions for further research, which is relevant with respect to these questions. For example, are companies participating in open source projects more successful than other companies, and what are the characteristics of companies participating in open source projects? It is worth noting that no identified paper presents much research about how companies' internal processes for collaborating with communities work. This could also be an area for further research.

Business models with open source in commercial organizations

Concerning the business models it is clear that companies involved in open source development besides developing open source products also offer customized software based on open source products. It is also common to offer consulting and training (e.g. [Bonaccorsi06]). It is also clear that business models include hybrid strategies, such as described by [West03], where the focus is on large software vendors. The paper presents an in-depth analysis on historical development of operating systems, computers, and business strategies adopted by vendors. It is also possible to base the business on being the link between an open source project and the enterprise customers by integrating the product in a commercial package [Munga09].

[Hauge09] presents a case study on a small Norwegian company that successfully established a business model around two open source products by establishing three specialized user communities. The paper also concludes that while it is important to attract developers to the community, it is as important to retain some control over the product for commercial benefits.

Bonaccorsi [Bonaccorsi05] investigates reasons why companies participate in open source communities. In particular, the paper analyzes discrepancies between attitudes and behaviors in relation to three primary research questions. The questions deal with motivation to set up an open source business, whether the firms' claims to uphold intrinsic, community-based values are aligned with the firms' actions, and finally, if there is discrepancy, are there any observable patterns. The conclusion points out that there is misalignment in attitudes and behavior of firms in open source market place, confirming earlier research that companies use intrinsic values to attract developers in order to fulfill their own extrinsic goals. In [Harison10] it is found that software companies with higher proportions of highly

educated personnel are more likely to adopt a business model based on supplying open source software.

[Westenholz06] offers an insight into challenges of creating a sustainable business around open source business model based on a case study. The study offers insight into shifting of business strategies conducted by the entrepreneur in order to make the business profitable around the combination of open source and proprietary software.

[Lindman09] asserts that business models can sometimes be too generic and undertake an exploratory case study of three different organizations in order to empirically identify different incentives companies have in releasing a product as open source beyond the revenue generating ones. The paper points to challenges in attracting and sustaining a community for a software product that is highly specialized.

Open source as part of component based software engineering

The article from Arhippainen [Arhippainen03] is a case study conducted at Nokia on the usage of the OTS components. The paper presents a detailed analysis on usage of third party components in general, and discusses advantages of using proprietary over open source components and vice versa. It also identifies issues related to software development methodology in terms of including third party components.

The research presented in [Ayala09] assesses the state of reusable components in the open source market based on survey conducted in Spanish and Norwegian companies. The results of the survey also assess the needs of OSS industrial users in component selection and identify challenges that can aid in maturing the open source components market.

The 5 articles [Li05a], [Li09], [Li05b], [Li06a], [Li06b] are based on the analysis of data collected through state-of-the practice survey conducted in Norway, Germany, and Italy. This research, conducted on over 100 projects that use proprietary or open source OTS components, looks into risks associated with use of such components, reasoning behind using OTS components, and impact on development process when using the OTS components. Some of the findings suggest that selection of OTS components is a very informal process, that OTS components are selected throughout the development process life cycle even though early selection yields benefits. It is also suggested that estimates on effort needed to integrate components is informal and dependent on experience, and as such, is often inaccurate. Furthermore, some general conclusions of the studies point that the OTS components rarely have negative impact on the system. Open source OTS components are used in the same manner as proprietary components, thus without modification. If a problem occurs with the OTS components, it takes substantial amount of effort to correct them.

It can be noticed that in [Li05a]–[Li09] there is no in-depth analysis of what kind of open source components as OTS components, were used by companies. For example, many mainstream proprietary IT workshops, sometimes very ”hostile” to the idea of using open source components, like the ones producing software for big financial houses, use PGP and other Unix based open source components as these over the time have become de-facto standard. Investigating into the diversity and type of open source OTS components that are used in projects can be a question for further research.

Using the open source process within a company

An interesting area that is investigated in [Gurbani05] and [Lindman08] is that of using an open source process within a company. That is, the product is not provided to any community outside the organization, but instead handled as an open source project within the company. One unit of the organizations owns the product and provides it to the rest of the organization. Everyone in the organizatin are allowed to use and modify the code, and changes are approved by the original owners as in any other open source project.

Gurbani et. al. [Gurbani05] presents a case study on transferring the open source development model for one software product at Lucent technology. In this case the approach was judged successful by the authors, for example because the product was needed in several products and the architecture was suitable for this. Lindman et. al. [Lindman08] also investigates the usage of an internal open source development methodology through a case study. This case study is conducted on usage of Nokia iSource portal for hosting projects. The portal became very popular for managing heterogeneous types of projects: SCM, distributed, agile, intercompany collaboration projects. The research results showed that implementation of open source project management tools can facilitate innovation within the company. In [Gaughan09] the area is also investigated in a set of studies. Positive aspects based on increased visibility in the organization such as as better code quality and pride in work are listed. However, visibility can also lead to other aspects, such as privacy and knowledge retention, and easier workplace monitoring. It should be noticed that all case studies are conducted at large companies, which probably is natural.

A number of further research questions can be identified. One concerns how contributions can be included in this kind of product when different developers have different needs for the developed product.

5 Discussion

In this review, 23 articles were identified. We do not think that this is a large number of articles compared to the importance of the field, and the general amount of discussion about how open source can be used by commercial organizations.

Many of the studies are in the form of surveys, which gives a broad and necessary understanding. Based on this it would probably be possible to conduct more studies investigating specific cases of implementation of methodologies for dealing with different aspects of open source in industry. More case studies could probably be conducted on all aspects of the research questions. More case studies could probably also provide more knowledge of research question 3 and research question 4. That is, research could be carried out to understand more about the cost and advantages of different approaches, and why different approaches are chosen. It is also worth noticing that there are no controlled experiments at all in the identified articles.

There is, of course, a risk that some articles have been missed in the search, either because the search string has not identified all relevant articles, or because the set of searched journals was not complete. The search string was developed through a ”trial and error” approach in order to find as many relevant articles as possible, but it is impossible to guarantee that all articles have been found. The same is true for the coverage of the search. It is not possible to guarantee that all relevant journals and conferences have been searched. Here the most severe risk is probably for articles not in the traditional software engineering literature, such as articles on business models, which is more general than traditional software engineering. Since there is a risk that all articles have not been found it is reasonable to discuss the effects of missing articles. Of course, the more complete the selection of articles is the better it is. However, in this case the objective is more to identify and summarize conducted research and experience than to carry out meta-analysis, which probably means that the effect of missing single articles is lower. Even if more articles would be found it is not unlikely that the major conclusions in terms of identified areas, and main conclusions in areas, would be the same.

6 Conclusions

Concerning research question 1 and 2, i.e. what approaches and process are used to introduce and provide open source components, it was possible to divide the the identified papers into different areas. The following areas were defined based on the articles: i) participation in open source development communities, ii) business models with open source, and iii) treating open source software as components in component based development. Besides this there are articles on iv) how open source processes can be used within a company.

In all areas experience in some form were presented in the articles, although the papers were on rather different areas. Some results were presented on motivation and incentives.

The areas are important for research and it is interesting to see that research is available in all these areas. The question of how to use open source practices within a closed company (iv) is for example an interesting area for further research.

Based on this review we also propose that further research is conducted on how companies can transform their proprietary software to open source and build a community on it. Further research related to all four research questions in Section 3.1 could involve more case studies on implementation of specific methodologies for dealing with different aspects of open source in industry.

Acknowledgment

This work was partly funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering, (`http://ease.cs.lth.se`).

References

[AB09] N. Abdelkafi and T. Blecker. "From open source in the digital to the physical world: a smooth transfer?" In: *Management Decisions* 47.10 (2009), pp. 1610–1632.

[CAC08] L. Chen, M. Ali Babar, and C. Cawley. "A status report on the evaluation of variability management approaches". In: *Proceedings of Evaluation and Assessment in Software Engineering (EASE)*. 2008.

[DGJ07] O. Dieste, A. Griman, and N. Juristo. "Developing search strategies for detecting relevant experiments". In: *Empirical Software Engineering* 14 (5 2007), pp. 513–539.

[DD08] Tore Dybå and Torgeir Dingsøyr. "Strength of evidence in systematic reviews in software engineering". In: *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*. ESEM'08. Kaiserslautern, Germany, 2008, pp. 178–187.

[FF02] Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison Wesley, 2002.

[Goo05] Sigi Goode. "Something for nothing: management rejection of open source software in Australia's top firms". In: *Information & Management* 42 (5 2005), pp. 669–681.

[HOA10a] Martin Höst and Alma Oručević-Alagić. "A Systematic Review of Research on Open Source Software in Commercial Software Product Development". In: *Proceedings of Evaluation and Assessment in Software Engineering (EASE)*. 2010.

[KC07] Barbara Kitchenham and S. Carter. *Guidelines for performing systematic literature reviews in software engineering, v. 2.3*. Tech. rep. Keele University and University of Durham, 2007.

[Kri06] Sandeep Krishnamurthy. "On the Intrinsic and Extrinsic Motivation of Free/Libre/Open Source (FLOSS) Developers". In: *Knowledge, Technology & Policy* 18.4 (2006), pp. 17–40.

[Lea+02] G. H. Leavesley et al. "A modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling". In: *Hydrological Processes* 16 (2 2002), pp. 173–187.

[LT01] Josh Lerner and Jean Tirole. "The open source movement: Key research questions". In: *European Economic Review* 45.4-6 (2001), pp. 819 –826.

[Ray01c] Eric S. Raymond. *The Cathedral and the Bazaar*. O'Reilly Media, 2001.

[RH09] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2 2009), pp. 131–164.

[SAB09] K-J. Stol and M. Ali Babar. "Reporting empirical research in open source software: the state of practice". In: *Proceedings International Conference on Open Source Systems*. Skövde, Sweden, 2009, pp. 156–169.

[SAB10] K-J. Stol and M. Ali Babar. "Challenges in using open source software in product development: a review of the literature". In: *Proceedings FLOSS*. Cape Town, South Africa, 2010, pp. 17–22.

[Web04a] S. Weber. *The Success of Open Source*. Harvard University Press, 2004.

Appendix: Articles included in review

Arhippainen03 L. Arhippainen, Use and integration of third-party components in software development. Technical report, VTT Publubilcation 489:84, 2003. In this report a case study on component based development, including the use of open source components, at Nokia is presented.

Ayala09 C. Ayala, Ø. Hauge, R. Conradi, X. Franch, J. Li, and K. Sandanger Velle, Challenges of the Open Source Component Marketplace in the Industry, In proc. OSS, pp. 213-224, 2009. The paper analyzes the state of open source market place and how companies interact to reuse components that the market place offers.

Bonaccorsi05 A. Bonaccorsi and C. Rossi, Intrinsic Motivations and Profit-oriented Firms in Open Source Software. Do firms practise what they preach?. In proc. OSS, pp. 241-245, 2005. The articles investigates true motivation behind companies involvement in open source activities based on data gathered through a survey of 146 Italian companies supplying open source solutions.

Bonaccorsi06 A. Bonaccorsi, S. Giannangeli, and C. Rossi Entry strategies under competing standards: Hybrid business models in the open source software industry. Management Science, 52(7):1085-109, 2006. This is a further analysis of the same survey as presented in [Bonaccorsi05]. They have developed a regression model explaining the "friendliness" to open source based on a set of factors.

Bonaccorsi07 A. Bonaccorsi, D. Lorenzi, M. Merito, and C. Rossi Business firms' engagement in community projects, empirical evidence and further developments of the research, In First International Workshop on Emerging Trends in FLOSS Research and Development, FLOSS'07, 2007. This is a survey based on a sample of 300 projects from SourceForge. In 97 of the projects there was at least one company participating. The three main types of involvement were "project coordinator", "collaboration", and "provision of code".

Gaughan09 Gaughan, G., Fitzgerald, B., and M. Shaikh, An examination of the use of open source software processes as a global software development solution for commercial software engineering. In proc. Euromicro Software Engineering and Advanced Applications, pp. 20-27, 2009. This paper summarizes experiences from using open source processes within companies.

Gurbani06 V.K. Gurbani, A. Garvert, and J.D. Herbsleb, A case study of a corporate open source development model. International Conference on Software Engineering, pp. 472-481, 2006. This paper presents a case study on transferring the open source development model for one software product to a commercial environment at Lucent technology, keeping the software proprietary in the company.

Harison10 E. Harison and H. Koski, Applying open innovation in business strategies: evidence from Finnish software firms. Research Policy, Vol. 39, pp. 351-359, 2010. A survey of 170 Finnish software firms with respect to business strategies is presented.

Hauge07 Ø. Hauge, C.F. Sørensen, and A. Røsdal, Surveying Industrial Roles in Open Source Software Development. In proc. OSS, pp. 259-264, 2007. This paper defines different industrial roles in open source community: provider, integrator, participant, and inner source software participant. Through a survey, it investigates motivation, challenges, and development practices of the companies taking on these roles within the ITEA COSI project.

Hauge09 Ø. Hauge and S. Ziemer, Providing Commercial Open Source Software: Lessons Learned. In proc. OSS, pp. 70-82, 2009. This paper presents a study on a small Norwegian software company that has built its business around own OSS products and compares the findings to other cases reported in literature.

Henkel08 J. Henkel, Champions of revealing – the role of open source developers in commercial firms, Industrial Corporate Change, Vol. 18, No. 3, pp. 435-471, 2008. This paper discusses how company employed persons can cooperate in open source communities, with focus on how code is committed to the community.

Li05a J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, An empirical study on off-the-shelf component usage in industrial projects. In Product Focused Software Development and Process Improvement (Profes), pp. 54-68, 2005. The paper presents survey conducted a large number of companies from Norway, Germany, and Italy on the off-the shelf (OTC) components usage. It

focuses on factors that influence the choice in terms of whether the OTS component is open source or proprietary.

Li05b J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, Validation of new theses on off-the-shelf component based development, International Software Metrics Symposium, pp. 231-240, 2005. The paper focuses on validating six theses related to usage of off-the-shelf components within companies.

Li06a J. Li, R. Conradi, O.P.N. Slyngstad, C. Bunse, U. Khan, M. Torchiano, and M. Morisio, An empirical study on decision making in off-the-shelf component-based development. In proc. International Conference on Software Engineering (ICSE), pp. 897-900, 2006. This article investigates research questions resembling those in [Li05a], but with a larger sample of projects.

Li06b J. Li, M. Torchiano, R. Conradi, O.P.N. Slyngstad, and C. Bunse, A state-of-the-practice survey of off-the-shelf component-based development processes. In Reuse of off-the-shelf Components, International Conference on Software Reuse, pp. 16-28, 2006. This paper focuses on the development process when OTS components are used.

Li09 J. Li, R. Conradi, C. Bunse, M. Torchiano, O.P.N. Slyngstad, and M. Morisio, Development with off-the-shelf components: 10 facts. IEEE Software, 26(2):80-87, 2009. This article basically summarizes the findings from the earlier articles by the same group of authors. The conclusions are presented in the form of 10 facts learned about development with OTS components.

Lindman08 J. Lindman, M. Rossi, P. Marttiin, Applying Open Source Development Practices Inside a Company. In proc. OSS, pp. 131-387, 2008. This paper investigates characteristics of using open source and agile development practices within a company. It argues that usage of such practices can unlock innovation potential within a company.

Lindman09 J. Lindman, J.P. Juutilainen, M. Rossi, Beyond the Business Model: Incentives for Organizations to Publish Software Source Code. In proc. OSS, pp. 47-56, 2009. The paper investigates incentives for companies to release software as open source through three exploratory case studies at different stages of code release.

Lundell06 B. Lundell, B. Lings, and E. Lindqvist, Perceptions and Uptake of Open Source in Swedish Organizations. In proc. OSS, pp. 155-163, 2006. This paper investigates usage of open source within Swedish companies from the perspective that goes beyond mere adoption of an open source software product. It focuses on participation of the companies within the open source communities in roles of code contributors on existing third party projects or its own products.

Munga09 N. Munga, T. Fogwill, and Q. Williams, The adoption of open software in business models: a Red Hat and IBM case study. In proc 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, pp. 112-121, 2009. This paper investigates the business models of open source related aspects of Red Hat and IBM.

Robles07 G. Robles, S. Dueñas, and J.M. Gonzalez-Barahona, Corporate Involvement of Libre Software: Study of Presence in Debian Code over Time. In proc. OSS, pp.

121-132, 2007. This paper investigates corporate involvement in Linux Debian GNU distribution over the period from 1998-2004 based on copyright attributions in the source code. The results of the research show that 6-7% of the code has been contributed by corporations.

West03 J. West, How open is open enough? melding proprietary and open source platform strategies. Research Policy, 32(7):1259-1285, 2003. The authors present a timeline for what has happened with respect to ”hybrid” software systems, that is software systems that consists of a mixture of open source software and proprietary software, such as an Apple computer. Three case studies are presented.

Westenholz06 A. Westenholz, Institutional Entrepreneurs and the Bricolage of Intellectual Property Discourses. In proc. OSS, pp. 183-193, 2006. This paper is a case study on an institutional entrepreneur who builds his own company on a business model that mixes open and closed source software practices.

USAGE OF OPEN SOURCE IN COMMERCIAL SOFTWARE PRODUCT DEVELOPMENT

PAPER II

Abstract

Open source components can be used as one type of software component in development of commercial software. In development using this type of component, potential open source components must first be identified, then specific components must be selected, and after that selected components should maybe be adapted before they are included in the developed product. A company using open source components must also decide how they should participate in open source project from which they use software. These steps have been investigated in a focus group meeting with representatives from industry. Findings, in the form of recommendations to engineers in the field are summarized for all the mentioned phases. The findings have been compared to published literature, and no major differences or conflicting facts have been found.

1 Introduction

Open source software denotes software that is available with source code free of charge, according to an open source license [FF02]. Depending on the license type,

Martin Höst, Alma Oručević-Alagić, and Per Runeson
International Conference on Product Focused Software Development and Process Improvement (PROFES), pp. 143-155, 2011

there are possibilities to include open source components in products in the same way as other components are included. That is, in a large software development projects, open source software can be used as one type of component as an alternative to components developed in-house or components obtained from external companies.

There are companies that have experience from using well known open source projects. Munga et al. [MFW09], for example, investigate business models for companies involved in open source development in two case studies (Red Hat and IBM) and concludes that "the key to their success was investing resources into the open source development community, while using this foundation to build stable, reliable and integrated solutions that were attractive to enterprise customers". This type of development, using open source software, is of interest for several companies. If open source components are used in product development there are a number steps that the company needs to go through, and there are a number of questions that need to be solved for each step.

First potential components must be identified, which can be done in several ways. That is the company must decide how to identify components. Then, when potential components have been identified, it must be decided which component to use. In this decision there are several factors to consider, and the company must decide how to make this decision. Using the components there may be reasons to change them, which gives rise to a number of questions on how this should be done and to what extent this can be recommended. A company working with open source components must also decide to what extent to get involved in the community of an open source project.

There is some research available in this area [HOA10a], although there is still a need to collect and summarize experience from companies working in this way. In this paper, findings are presented from a workshop, in the form of a focus group meeting, where these topics were analyzed by industry representatives.

The outline of this paper is as follows. In Section 2 the methodology of the research is presented, and in Section 3 the results are presented. The results are compared to results presented in the literature in Section 4, and in Section 5 the main conclusions are presented.

2 Methodology

2.1 Focus group

The workshop was run as a focus group meeting [Rob02; KBL08]. At the workshop, participants informally presented their experience from development with open source software, for example from using open source components in their product development, or from participating in open source communities. The intention was to give all participants both an insight into how others in similar situ-

ations work with these issues, and to get feedback on one's own work from other organizations. The result of a similar type of workshop was presented in [ER10].

Invitations to the workshop were sent to the network of the researchers. This includes earlier participants at a seminar on "research on open source in industry" where rather many (≈ 50) people attended, and mailing lists to companies in the region. This means that the participants cannot be seen as a representative sample of a population and generalizations cannot be made in traditional statistical terms. Instead analysis must be made according to a qualitative method, e.g. as described by Fink [Fin02, p. 61-78]. This is further discussed in Section 2.4.

2.2 Objectives and discussion questions

The main research questions for the study were:

- How should open source components for inclusion in products be selected? Is there a need to modify selected components, and if so, how should this be done?

- To what extent is code given back to the open source community, and what are the reasons behind doing so?

Discussion questions could be derived from the objectives in different ways. One possibility would be to let the participants focus on a specific project and discuss issues based on that. The advantage of this would be that it would probably be easy for the participants to know what actually happened since it concerns a specific project. The difficulties with this approach are that there is a risk that participants have valuable experience from more than one project and therefore cannot express all experiences they have since they should focus on one specific project. There is also a risk that data becomes more sensitive if it is about a specific project. Another alternative is to ask about more general experience from the participant and let them express this in the form of advice to someone working in the area. That is, the participants use all the experience and knowledge they have, without limiting it to a specific project or presenting details about projects, customers, etc. This was the approach that was taken in this research.

Based on the objectives of workshop, the following discussion questions were phrased:

1. How should one identify components that are useful, and how should one select which component to use?

2. How should one modify the selected component and include it in ones product?

3. How should one take care of updates from the community?

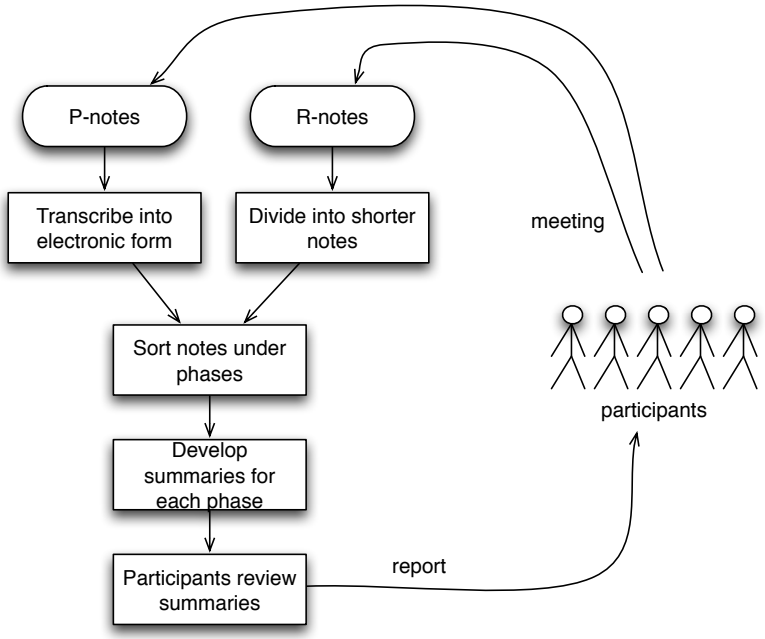

Figure 1: Main analysis steps

4. How should one handle own modifications/changes? What are the reasons for giving back code (or not giving back code)?

In order to get a good discussion, where as many relevant aspects as possible were covered, it was monitored in the following way. For each discussion question, the participants were given some time to individually formulate an answer, or several answers, on a Post-it note. When individual answers had been formulated each participant presented their answer to the others, and the notes were posted on the wall. During the discussions, the researchers also took notes.

2.3 Analysis procedure

The main data that was available for analysis were the notes formulated by the participants (”P-notes” below) and the notes taken by the researchers (”R-notes” below). The analysis was carried out in a number of steps, which are summarized in Figure 1 and explained below.

First all P-notes were transcribed into electronic form. In this step one note was transformed into one line of text. However, in some cases the participants wrote lists with more than one note at each piece of paper. In these cases this was clearly

marked in the transcript. When interpreting the notes, the researcher were helped by the fact that the participants had presented the notes at the meeting earlier.

The R-notes were derived by dividing a longer text into single notes. After this the P-notes and the R-notes were on the same form.

After this a set of phases were defined, based on the lifecycle phases in software development. These phases were based on the areas covered by the questions, but not exactly the same. Then, all notes could be sorted under the phases in which they are relevant.

Next, all notes were grouped in related themes within phases, and based on these summaries were developed. This means that one presentation summary was developed for each phase. The final version of these summaries are presented in Section 3.

Based on this, a report was developed with the summaries. The participants were given the possibility to review and adapt the summaries in the report. This resulted only in minor changes.

This procedure results in a summary, as presented in Section 3. The results were given back to the participants in the form of a technical report. This result is also compared to the literature in Section 4 of this article.

2.4 Validity

Since the collected data is analyzed qualitatively, the validity can be analyzed in the same way as in a typical case study, which in many cases also is analyzed qualitatively. Validity can for example be analyzed with respect to construct validity, internal validity, external validity, and reliability [Rob02; RH09].

Construct validity reflects to what extent the factors that are studied really represent what the researcher have in mind and what is investigated according to the research questions.

In this study we believe that the terms (like ”open source”, ”component”, etc.) that are used are commonly used terms and that the risk of not meaning the same thing is low. It was also the case that the participants formulated much of the notes themselves, which means that they used terms that they fully understood. Besides this, the researchers participated in the whole meeting, which means that it was possible for them to obtain clarifications when it was needed. Also, the report with the same material as in Chapter 3 of this paper was reviewed by the participants.

Internal validity is of concern when causal relations are examined. In this study no causal relations are investigated.

External validity is concerned with to what extent it is possible to generalize the findings, and to what extent the findings are of interest to other people outside the investigated case.

The study was conducted with a limited set of participants from a limited set of organizations. This means, of course, that the results cannot automatically be generalized to other organizations. Instead it must be up to the reader to judge if it is reasonable to believe that the results are relevant also for another organization or project. The results are compared and validated to other literature and the type of results is not intended to be specific for a certain type of results.

It should also be noticed that the findings from the focus group are based on the opinions of the participants. There may be a risk that the opinions are very specific for one participant or for the organization he/she represents. The nature of a focus group meeting helps avoiding this problem. According to Robson there is a natural quality control and participants tend to provide checks and react to extreme views that they do not agree with, and group dynamics help in focusing on the most important topics [Rob02, Box 9.5].

Reliability is concerned with to what extent the data and the analysis are dependent on the specific researchers.

In order to obtain higher validity with respect to this, more than one researcher were involved in the design and the analysis of the study. Also, as mentioned above, the report with the same material as in Chapter 3 of this paper was reviewed by the participants.

Another aspect that is relevant to this is how the questions were asked and what type of data the participants are asked to provide. In order to avoid problems with confidentiality, the participants were asked to formulate answers more as advice to someone who is working in the area than as concrete experiences from specific (and named) projects. We believe that this makes it easier to provide data for this type of participants.

3 Results from focus group meeting

3.1 Participants

At the workshop the following participants and organizations participated:

A. Four researchers in Software Engineering from Lund University, i.e. the authors of this paper and one more person

B. One researcher in Software Engineering from another university

C. Two persons from a company developing software and hardware for embedded systems.

D. One person from a company developing software and functionality based on an embedded system

E. One person from an organization developing software and hardware for embedded systems with more than 10 years tradition of using open source software

F. One person from an organization with the objective of supporting organizations in the region to improve in research, innovation and entrepreneurship in mobile communications

That is, in total 10 persons participated, including the authors of this paper.

3.2 Identification

Previously, companies were used to choose between making components themselves or to buying them. Now the choice is between making or buying, or using an open source component. That is, there is one more type of component to take into account in the identification process. It should also be pointed out that it is a strategic decision in terms of whether the product you are developing should be seen as a closed product with open source components or as an open source product.

When components are identified it is important that this is based on a need in the development and that it maps to the product requirements. When it comes to the criteria that are used when identifying components, they should preferably be identified in advance.

In the search process, the advice is to start with well-known components and investigate if they fulfill the requirements. There is also a lot of knowledge available among the members in the communities, so if there are engineers in the organization that are active in the community, they should be consulted. A further advice is to encourage engineers to participate in communities, in order to gain this kind of experience. However, the advice to consult engineers in the organization is not depending on that they are members of the communities. A general knowledge and awareness of existing communities is also valuable.

The next step is to search in open source forums like sourceforge and with general search engines like google. The advice here is to use technical terms for searching (algorithm, protocols, standards), instead of trying to express what you try to solve. For example, it is harder to find information on "architectural framework" than on specific techniques for this.

3.3 Selection

The more general advises concerning the selection process is to, again, use predefined criteria and recommendations from colleagues. It is also possible to conduct a basic SWOT-analysis in the analysis phase.

A more general aspect that is important to take into account is if any of the identified components can be seen as an ”ad hoc standard”, meaning that they are

used in many products of that kind and if it will increase interoperability and the ease communication with other components. One criterion that is important in this selection concerns the legal aspects. It is necessary to understand the constraints posed by already included components and, of course, other aspects of the licenses.

Other more technical criteria that are important include programming language, code quality, security, and maintainability and quality of documentation. It is necessary to understand how much effort is required to include the component in the architecture and it is necessary to understand how the currently used tool chain fits with the component. A set of test cases is one example of an artifact that is positive if it is available in the project.

A very important factor concerns the maturity of the community. It is necessary to investigate if the community is stable and if here is a ”backing organization” taking a long-term responsibility. It is also important to understand what type of participants in the community that are active. The roadmap of the open source project is important to understand in order to take a decision that is favorable for the future of the project.

3.4 Modification

First it should be emphasized that there are disadvantages of making changes to an own version of the components. The disadvantages are that the maintenance costs increase when updates to new versions of the components are made, and it is not possible to count on extensive support for specific updates from the community. So, a common recommendation is to do this only if it is really necessary.

There are some reasons why modifications must be made. Especially adaptation to specific hardware is needed, but also optimizations of different kind. When these changes are made it is in many cases favorable to give back to the community as discussed in the next section but if this is not possible an alternative is to develop ”glue software” and in that way keeping the API unchanged.

If changes should be made it is necessary to invest effort in getting a deep knowledge of the source code and architecture, even if a complete set of documentation is not available.

3.5 Giving back code

It is, as discussed in the previous section, in many cases an advantage to commit changes to the open source project instead of working with an own forked version. In this way it is easier to include updates of the open source component. In order to manage this it is in many cases an advantage to become an active member of the community, and maybe also take a leading role in it. When modifying an open source component it is, of course, an advantage if ones own changes can be aligned with the future development of the open source component.

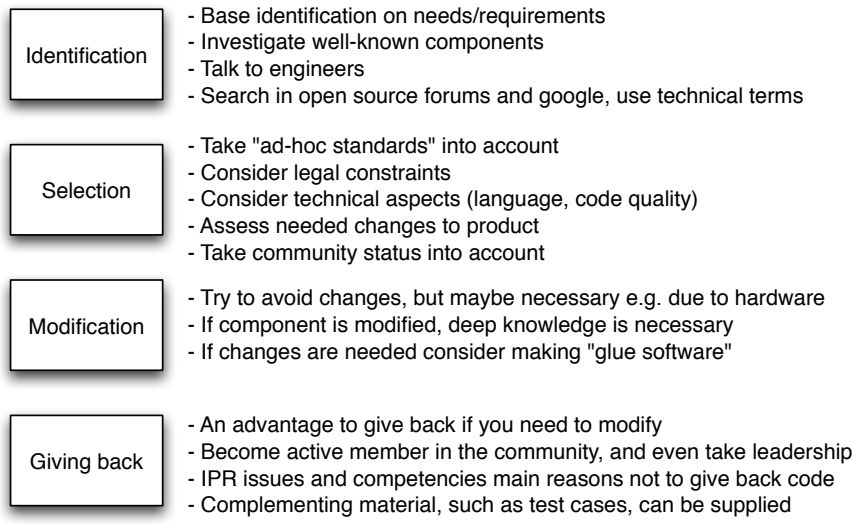

Figure 2: Main findings from workshop

However, there are some reasons not to give back changes too. The most important reason is probably that you want to protect essential IPR's and core competences in the organization. That is, key competence must in some situations be hidden from competitors. It should, however, be noticed that there may be requirements from the license to give back code. Also, after some time, all software will be seen as commodity, which means that this kind of decision must be reconsidered after a while. Another reason not to make changes public is that possible security holes can be made public. In some cases it is easier to get a change accepted if test cases are supplied.

3.6 Summary of results

The main findings from the workshop, in terms of recommendations for the four phases, are summarized in Figure 2.

4 Conclusions

We believe that many of the recommendations from the participants are important to take into account in research and in process improvement in other companies. The most important findings from the workshop are summarized below. The find-

ings are in line with presented research in literature as described in Section 4, although the details and formulations are specific to the results of this study.

In the identification phase it is important to take the needs and the requirements into account, and to investigate well-known components. It is also advised to discuss the needs with engineers in the organization, since they can have knowledge of different components and communities. When forums are searched, an advice is to use technical terms in the search string. When selecting which components to use it is important to, besides taking technical aspects, like programming language, into account, also consider legal constraints and ”ad-hoc standards”. It is important to investigate the status of the community of a project, and the future of the project, which for example depends on the community. In general it can be said that changing components should be avoided if possible. If it is possible to make adaptations with ”glue-code” this is in many cases better since less effort will be required in the future when components are updated by the community. However, there are situations when it is necessary to make changes in the components.

Even if there may be issues with property rights, it is in many cases an advantage to provide code to the community if changes have been made. In general it can be said that it is advised to become an active member in open source projects.

The findings from the focus group meeting were compared to published literature, and no conflicting facts were found.

Together with further research on the subject it will be possible to formulate guidelines for software project managers on how to work with open source software.

Acknowledgments

The authors would like to thank the participants for participating in the study.

This work was funded by the Industrial Excellence Center EASE – Embedded Applications Software Engineering, (`http://ease.cs.lth.se`).

References

[ER10] Emelie Engström and Per Runeson. “A Qualitative Survey of Regression Testing Practices”. In: *Proceedings of International Conference on Product-Focused Software Process Improvement (PROFES)*. 2010, pp. 3–16.

[FF02] Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison Wesley, 2002.

[Fin02] Arlene Fink. *The Survey Handbook*. 2:nd. Sage Publications, 2002.

[HOA10a] Martin Höst and Alma Oručević-Alagić. "A Systematic Review of Research on Open Source Software in Commercial Software Product Development". In: *Proceedings of Evaluation and Assessment in Software Engineering (EASE)*. 2010.

[KBL08] Jyrki Kontio, Johanna Bragge, and Laura Lehtola. "The Focus Group Method as an Empirical Tool in Software Engineering". In: *Guide to Advanced Empirical Software Engineering*. Ed. by Forrest Shull, Janice Singer, and Dag I. K. Sjøberg. Springer, 2008.

[MFW09] Neeshal Munga, Thomas Fogwill, and Quentin Williams. "The adoption of open source software in business models: A Red Hat and IBM case study". In: *Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists,* 2009, pp. 112 –121.

[Rob02] Colin Robson. *Real World Reserach*. 2:nd. Blackwell Publishing, 2002.

[RH09] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2 2009), pp. 131–164.

A Case Study on the Transformation From Proprietary to Open Source Software

PAPER III

Abstract

Many large companies, from the traditionally proprietary software industry, are opening up and embracing the open source software (OSS) development process model as a part of their business strategy. Despite the recognized potential the OSS community offers, there are still many questions and unknowns about the transition process. We present an extensive analysis of static software quality metrics changes for Ingres, an open source enterprise database management system (DBMS), as the software was moved from the proprietary into open source software development environment. The software quality metrics of special interest for the research are cyclomatic complexity, effective lines of code, the degree of system modularity, and the amount of comments in the code. The conducted research shows an overall improvement in the software quality metrics and significant increase of the source code base. The overall improvement is comprised of a decrease in software quality metrics for source files that were changed between the proprietary and the OSS version and an increase in software quality metrics for the source files added through Ingres OSS community development process.

Alma Oručević-Alagić, Martin Höst
Extended version of 6th International IFIP WG 2.13 Conference on Open Source Systems (OSS 2010), Notre Dame, IN, USA, May 30 - June 2, 2010, Proceedings, pp. 367-372, 2010

1 Introduction

In the last few decades, a traditional software production has come to presume an ”in-house” or closed source development process, which means that all development is carried out by engineers employed within the same organization and the software source code that is not available for the general public. There are several different types of OSS licenses [FF02; Ray01a] with common characteristic of source code being available free of charge to the general public. This means that the traditional business models which are based on the sale of software licenses are not applicable for software produced through the OSS community process.

The traditional software development has been taking advantage of third party components, among which open source components have been playing an important role [Li+06b; Li+06a]. Thus, the open source components are perceived as just another third party component or an alternative to in-house solution and COTS. This approach has large similarities to traditional development, but a difference is that the company that uses the open source component must decide to what extent it will participate in the development of the component. It would, of course, be possible to use the available component as is, but for several reasons companies may want to participate in OSS development process, e.g. to increase their understanding of the software they are using, to be able to affect the software evolution, and to support it in order to secure its future existence. There is also an interest not only in participating in an open source community, but to provide a software product to an open source community, e.g. [Bon+07].

Since virtually no development starts from scratch, an important process to investigate is that of transforming a traditional proprietary software product to an open source product. In this process the company will probably have less control over the evolution of the software than in traditional development, and an important question concerns what impact the changes made by the community will have on the software quality metrics such as cyclomatic complexity of the code and modularity.

The purpose of this paper is to investigate one case of this type of transformation. The case that is chosen is the Ingres database management system [Ing09], which, according to many, has received a new breath of life after its release into the open source community. The software was for a long time proprietary and after that it was transformed to open source. Software metrics that show how the source code has evolved under the OSS development process, are identified and then analyzed for this product before the transition started and after the transition has occurred.This is one case of this type of transformation and the intention is to learn as much as posible from this case. It should however be seen as one case and the results are not by default representative for all other cases. Instead, other case studies can bring light on other cases and together they can form aggregated knowledge.

The outline of this paper is as follows. In Section 2, the background informa-

tion on the case software and related research studies is presented. In Section 2, the research method is further defined. Section 3.2 presents the obtained results, while Section 4 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 5.

2 Background and related work

The roots of the Ingres reach back to the 1970s and UC Barkley, when the initial development of the software was started as open source. The same code base was modified and spawned into Sybase and Microsoft SQL server in 1980s. In 1994, the software was acquired by CA (Computer Associates) from the ASK Group, the company that created a proprietary version of the Ingres code. By the year 2004, there were roughly around 5000 customers of the proprietary Ingres DBMS software which was rather small customer base compared to the customer base of some of its major competitors such as MySQL and Oracle. In order to increase the market share, CA decided to transform the product to open source in 2004. The company implemented loss-leader/market positioner business model [Ray01a]. To reaffirm its commitment and support to Linux development process, Computer Associates has contributed Kernel Generalized Event Model software to Linux. The software, incorporated into the Linux kernel, improves security of Linux and feeds performance information from Linux systems to management systems [O'G04].

In November of 2005, Computer Associates and Garnett and Helfirch capital created a new company, Ingres Corporation. The main role of Ingres Corporation is to oversee the open source development process, provide support and services for Ingres and OpenRoad. Today, Ingres customer base includes 10,000 enterprise customers, among which 136 belong to the Fortune 500 companies like 3M, Bea Systems, and Lufthansa [Ass09]. Hence, the positive turnaround Ingres has made since it went open source, make the analysis of the software code quality metrics even more interesting, especially when viewed from the historical perspective, i.e. comparing the code metrics of the last proprietary version of the software from 2004, and the most recent one released as open source in November of 2008.

The Open Source Report, released in 2008, is the product of a two years long effort by Coverity Software with support from the US Department of Homeland Security[Sof08]. Over 55 million lines of code over the two year period for more than 250 open source projects, totaling around 10 billion lines of code, was analyzed. One of the main purposes of this study was to provide developers with better understanding of the relationships between software defects and fundamental elements of coding such as function lengths and code complexity.

Bonaccorsi et. al. [Bon+07] have investigated business firms involvment in OSS projects. They found that in 97 out of 300 sampled projects, at least one business firm was involved. Three main kinds of involvement were found: project

coordination, which was the most frequent case, collaboration of software development, and provision of code. This confirms the need for investigation of the process of transforming proprietary software to OSS.

Stemlos [IS02] conducted code quality analysis in open source development for 100 applications written for Linux. It was determined that some open source products have lower quality of code produced in OSS environment then that which is expected as an industry standard. Unlike this research which compares software quality of the same product in its latest proprietary version and version created after 4 years of OSS development, the presented work in [IS02] analyses quality metrics for code produced by OSS community against industry standard.

Related work also includes the work of Gurbani et. al. [VKG06], who have conducted a case study of managing an internal project as an OSS project. The difference compared to this research is that the work presented in [VKG06] concerns a project that was kept within a company, and there was not the same focus on code quality metrics.

3 Research approach

3.1 Introduction

The study is conducted as a case study [RH08]. The investigated case is the transformation of the Ingres code, from proprietary to open source. The study is exploratory with the overall objective to understand what type changes that were made to the case software and how this affected some commonly use code metrics.

In this study a quantitive approach has been taken. Hence, code metrics such as cyclomatic complexity, effective lines of code, modularity or average file function count, have been measured and compared. For example, the study performed by Zhang [HZ07] on public NASA datasets shows that static code complexity measures can be useful indicators of component quality.

In order to analyze and compare code metrics of the most recent proprietary version, further referred as 2004v, and open source version, further referred as 2008v, of Ingres, the 2004v was obtained by directly contacting the Ingres Corporation. The 2008v was downloaded from the Ingres Open Source community web site[1] in November of 2008.

3.2 Research questions

The following research questions were investigated during the research:

1. What parts of the Ingres DBMS software components went through the most source code changes in terms of source files added, changed, and deleted?

[1] http://community.ingres.com/wiki/Ingres _DBMS _Downloads

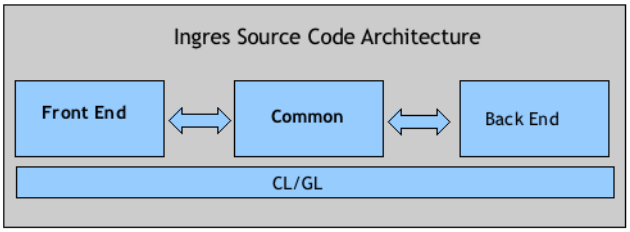

Figure 1: High Level View of Major Source Code Components of Ingres Source Code Architecture

2. How did Ingres DBMS code base change under the OS community process in terms of static source code metrics?

For research question 1, the focus is on architecture level changes. For research question 2, metrics with respect to quality attributes like size, complexity, and amount of comments in the code are of interest. It should be noted that there is no simple linear relationship between these metrics and how ”good” the software is. For example, with respect to complexity it is in general recommended not to have too high complexity, but when the complexity is below a certain value, required functionality cannot be implemented. For comments there is probably a similar relationship. If there are very few comments it is probably not as good as if there are more comments, but if there are very many comments it is probably not better than, or even as good as, if here a bit fewer comments. This makes it important to highlight the objective of this study, i.e. to understand what changes that have been carried out, and not to assess or compare the case software to any other software.

3.3 Investigated software

In order to ease the understanding of the approach for collecting and analyzing data the high level architecture of the case software is described here. The architecture is illustrated in Figure 1. It is grouped into four major components:

Front End: Functionality covers user interface facilities.

Back End: Functionality covers DBMS server functionality.

Common: Functionality covers connectivity and communications between the front end the back end.

Utility: Functionality covers utility libraries that interact with operating system.

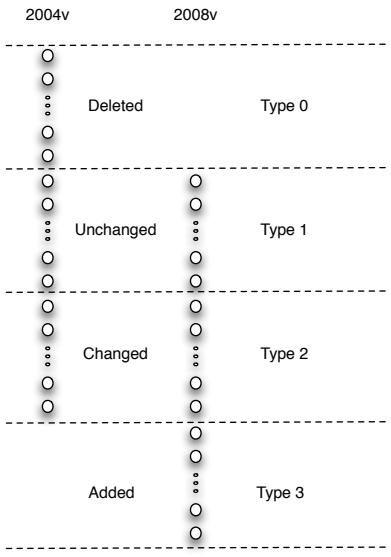

Figure 2: File types (files represented by circles)

A program that parses through the 2004v and 2008v code base was created, or more specifically, the files and subdirectories under the main `src` directory that contains all of the source files. The files were compared between the two code bases in order to determine which files exist only in 2004v, further referred as file type 0, which files are identical in 2004v and 2008v versions, further referred as file type 1, which files were modified between the two versions, further referred as file type 2, and finally, which files only exist, in 2008v, further referred as file type 3. The following list provides an overview of the changes, and the types are also shown in Figure 2.

- File type 0 : Source files that can be found only in 2004v
- File type 1 : Source files identical - unchanged between the 2004v and 2008v
- File type 2: Source files that were changed between the 2004v and 2008v
- File type 3: Source files that were added in 2008v

3.4 Metrics

The following metrics were measured in both versions:

- lines of code (LOC)
- effective lines of code ($ELOC$)
- comment lines (C)
- total cyclomatic complexity (TCC)
- file functions count (FFC)

All metrics are calculated on file level. For example, LOC denotes how many lines of code there are in each source file, and the sum of these values for all source files denotes the total number of lines of code per source code base. The LOC metric takes into consideration all lines of code but blank only or comment only lines. Hence, all lines in the source file except the blank lines or comment lines are taken into consideration by the LOC statistic.

For coding purposes developers often use braces or parenthesis to make code more readable, but this practice can inflate LOC metrics [RSM08]. The $ELOC$ metric takes into consideration all lines of code except blank only or comment only lines as well as the lines containing only standalone braces or parenthesis (`{`, `}`, `(`, `)`) Thus, lines counted by the $ELOC$ metric are a subset of the lines counted by the LOC metric.

C denotes the number of comment lines. The comment lines can appear by themselves on one physical line of code, or can be co-mingled.

The TCC or total cyclomatic complexity metric, also known as McCabe's cyclomatic complexity, is the degree of logical branching per source file. The cyclomatic complexity is calculated as

$$TCC = E - N + 2P$$

where E denotes the number of edges of the graph, N the number of nodes of the graph, and P is the number of connected components [FP98]. This value is calculated for each function in a file. For each file a value is calculated as the sum of the complexity of each function in the file.

FFC, or total number of file functions, within a source file determines the modularity of the file.

The FFC metric combined with $ELOC$ metric produces average number of effective lines of code $AELOC$ metric, calculated as:

$$AELOC = \frac{ELOC}{FFC}.$$

In the same way, the average cyclomatic complexity (ACC) can be calculated as:

$$ACC = \frac{TCC}{FFC}$$

In addition to the above metrics, the amount of comments are of interest. Therefore a metric describing the relative number of comments in each file RC is calculated:

$$RC = \frac{C}{ELOC + C}$$

Metrics for each file were derived with a metrics tool and stored in a database together with file type information for analysis.

3.5 Analysis procedure

Analysis with respect to research question 1 was conducted by determining the percentage changes in terms of file type 0, file type 1, file type 2, and file type 3 per major components of the source code. The components correspond to directories listed under "src" directory which houses all of the Ingres source code. In addition, more granular analysis were conducted on file level.

When analysing research question 2, the differences between the different versions of the case software were investigated with hypothesis tests. The null hypotheses state that the code changes made to 2004v, resulting in 2008v, had no impact on code metrics. The two-sided alternative hypotheses state that there was an impact.

Let $T = \{0, 1, 2, 3\}$ denote file types according to above and let $M = \{AELOC, ACC, RC\}$ denote the different metrics of interest, so that $\mu_m(v, T_s)$ represents the expected mean of metric $m \in M$ for all files of types $T_s \subseteq T$ in version v. Then the following null hypotheses and alternative hypotheses have been defined:

$$H0_{m,changed} : \mu_m(2004v, \{2\}) = \mu_m(2008v, \{2\})$$

$$Ha_{m,changed} : \mu_m(2004v, \{2\}) \neq \mu_m(2008v, \{2\})$$

and

$$H0_{m,new} : \mu_m(2004v, \{0, 1, 2\}) = \mu_m(2008v, \{3\})$$

$$Ha_{m,new} : \mu_m(2004v, \{0, 1, 2\}) \neq \mu_m(2008v, \{3\})$$

and

$$H0_{m,all} : \mu_m(2004v, \{0, 1, 2\}) = \mu_m(2008v, \{1, 2, 3\})$$

$$Ha_{m,all} : \mu_m(2004v, \{0, 1, 2\}) \neq \mu_m(2008v, \{1, 2, 3\})$$

That is, three null hypotheses have been formulated for each metric in M so that there is one concerning only the changed files ($H0_{m,changed}$), one concerning all files from 2004v and only newly added files to 2008v ($H0_{m,new}$), and one concerning all the files in 2004v and 2008v ($H0_{m,all}$). This means that $|M| \times 3 = 3 \times 3 = 9$ null hypotheses and equally many alternative hypotheses have been defined in total.

This means that hypothesis tests are conducted by treating file level measurements as independent samples. This gives the possibility to see if observed changes are of a true pattern (it is possible to reject the null hypothesis) or if they have occurred more by chance (it is not possible to reject the null hypothesis). Analysis of data for distributions of metrics results for version 2004v and 2008v were performed and it was determined that data for the metrics do not follow normal distribution. Hence in order to compare distribution of the metrics, non parametric tests, Mann-Whitney and Wilcoxon were performed. The Wilcoxon Signed-Rank Test for matched pairs was used in order to compare paired data sets (i.e., in analysis of $H0_{m,changed}$), and the Mann-Whitney U test was used to compare un-paired data (i.e., in analysis of $H0_{m,all}$ and $H0_{m,new}$).

3.6 Validity

In this section the validity of the research is analysed with respect to the types of validity threats presented, for example, in [Woh+00].

Construct validity: The construct validity is related to the relationship between the concepts and theories behind the experiment and what is measured and affected. Commonly accepted metrics for static software quality measurements such as cyclomatic complexity, lines of code, file function count, effective lines of code, were used. This means that the risk of using metrics that do not represent the concept of code quality is lowered.

Conclusion validity: The conclusion validity is concerned with the possibility to draw correct conclusions regarding the relationship between treatments and the outcome of an experiment. All of the population distributions analyzed did not follow normal distributions, and thus, in order to analyze the distributions, tests of lower statistical power than the t-test had to be used. Hence, the statistical tests used to analyze the data were Wilcoxon Signed-Rank Test for a matched pairs experiment and the Mann-Whitney U Test for independent random samples. This means that the statistics were not dependent on a normal distribution. It should be noted that the number of data points can be considered high. This means that even if non-parametric tests were to be used, the chance of detecting differences in distributions can be seen as high due to the large amount of data.

Internal validity: The internal validity is concerned with factors that may affect the dependent variables without the researcher's knowledge. Over some period of time, software quality will change, as the software goes through various maintenance processes. The software that is left unchanged for a longer period of time, will normally see its software quality metrics depreciated e.g. as stated by Lehman's laws [Som07]. On the other hand, a software that is maintained will see a change in its software quality metrics. It is unknown

what an average change in software quality metrics for similar products over four year period would be in proprietary environment. If the average change amount was established then it could be compared to the one introduced by the OSS maintenance process. However, the fact that the product has been transformed to OSS has been a major event for the product during these years, and it is not probable that the transformation have not had any affect on the quality.

External validity: The external validity is related to the ability to generalise the results of the experiments. While the case software is quite relevant in terms of its source code size, market, decades long life span, and impressive customer base, more research is needed to make general conclusion on whether the results of this study can be applied to other similar software systems. This is a case study, and the focus is on the case as such and not on generalization.

4 Results

4.1 Research question 1: Distribution of source code changes

Figure 3 shows the distribution of 2008v source files in percentages for each subdirectory under src directory, or i.e. directories: src/tst, src/tools, src/testtool, src/sig, src/ha, src/gl, src/front, src/dbutil, src/common, src/cl, src/back, and src/admin. Not all of the source code subdirectories will be analyzed in more detail, but only front, back, common, gl, and cl, since these directories contain almost 95% of the code. Hence, the most of the source files are located under /src/front directory or 54.7% of all 2008v. In the second place is src/cl directory housing 15.40% of source files in 2008v, followed by the src/back and src/common, housing 14.06% and 10.44% of all 2008v source files, respectively. Thus, these four directories contain 94.6% of 2008v source files. For this reason, in the following discussions, the types of changes made to these directories are analyzed in more detail (see section 2.3).

Under the src/front directory the components that belong to the front end layer of the software are stored. The front end functionality includes embedded SQL support, character based tools such as Application by Form (ABF), Query by Form (QBF), Report by Form (RBF) and Terminal Monitor (TM). Furthermore, the front end also includes Web Deployment Option that enables inclusion of data from Ingres data source into HTML page. Finally, the front end also houses the functionality related to replication that facilitates consistency of data sources located on different targets. From the data in Figure 3, it is clear that over 50% of all changes in the front end layer are due to the addition of the new source file components (type 3). Another 33% of changes are due to changes (file type 2).

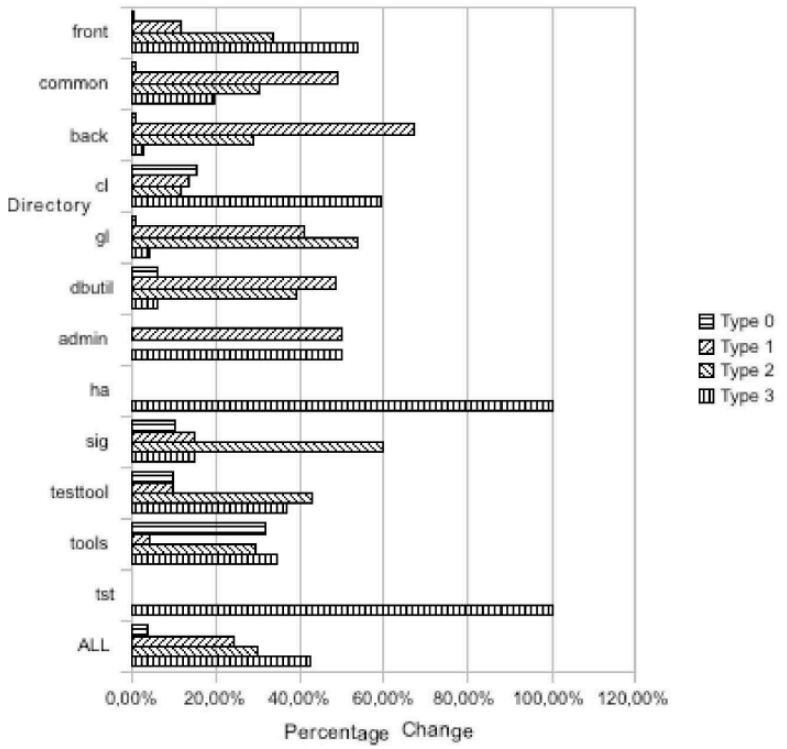

Figure 3: Distribution of file types (0, 1, 2, 3 according to Section 2.3) for source code directories

Therefore, around 88% of front end layer source files have been changed since the case software went open source. When this fact is combined with the fact that the front layer houses 54.7% of all source files, it can be said that some 48%, or almost a half of the source code was added and changed, with 44% of all 2008v file changes being contributed to addition of new source files (file type 3) to the front layer.

Under src/cl library source files for Ingres Compatibility Library are housed. The Ingres Compatibility Library provides interface to underlying operating system. This library provides the common interfaces for Memory, I/O, IPC and it may not call the higher levels of Ingres code. Referring to Figure 2.0 it can be observed that this library grew 69% between 2004v and 2008v, that is it contains 69% of file type 3 files. The src/back end components are deemed very important as the proper functioning of these components significantly affects database performance. The back end components are responsible for query storage, parsing, optimization and execution. In addition, the back end also facilitates logging, locking, archiving and

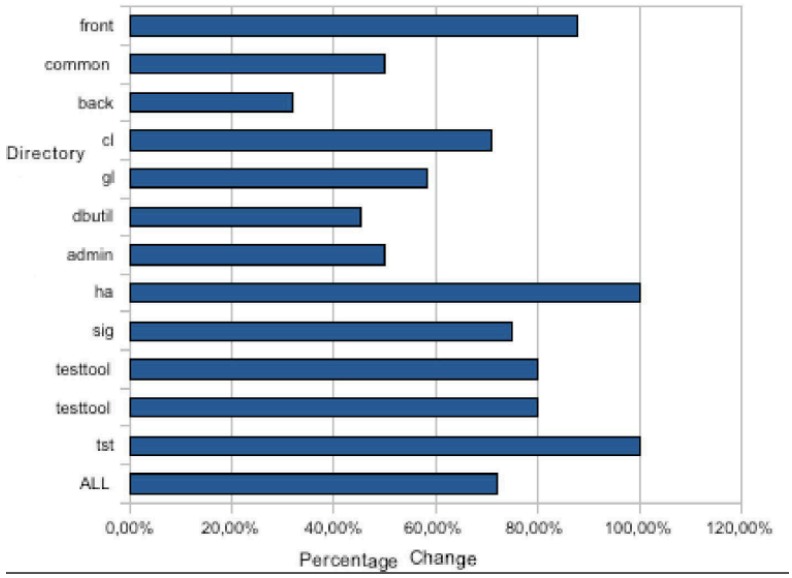

Figure 4: Total source code changes (File Type 2 and File Type 3) per each top level source code directory

recovery operations. The back end components went through the least amount of source code changes and additions, having 67.5% of code unchanged (file type 1) between the 2004v and 2008v. It also contains the least number of file additions (file type 3), thus only having 2.92% of the total number of the source files added (file type 3).

Finally the src/common contains components used by both, front and back end. These components include Abstract Datatype Facility (ADF), Common Utility Facility (CUF), General Communications Facility (GCF), Ingres .NET Data Provider, Java Database Connectivity (JDBC), Open Database Connectivity (ODBC) and Open Application Program Interface (Open API). The common components contain 49.05% of file type 1, or almost half of its components are same for 2004v and 2008v. It can be observed that 19.5% of its file were of file type 3, or newly added components.

4.2 Research question 2: Change in Static Code Quality Metrics

Table 2 displays code metric statistics summarized for the entire source code base of 2004v and 2008v. Hence, it can be observed that the number of file functions, lines of code and effective lines of code has increased. As one would expect,

Table 1: Summary of source code metrics for the whole system

Code Metric	2004v	2008v
Total LOC	840,502	1,442,225
Total $ELOC$	650,055	1,110,261
Total C	484,349	630,635
Total TCC	167,753	300,493
Total FFC	15,588	45,216

Table 2: Mean values and p-values

$H0$	mean 2004	mean 2008	p	reject $H0$
$H0_{AELOC,changed}$	41.35	41.69	< 0.001	yes
$H0_{ACC,changed}$	10.47	10.80	< 0.001	yes
$H0_{RC,changed}$	0.53	0.54	1	no
$H0_{AELOC,new}$	23.68	11.85	0.0042	yes
$H0_{ACC,new}$	6.12	2.80	0.01	yes
$H0_{RC,new}$	0.56	0.42	< 0.001	yes
$H0_{AELOC,all}$	23.68	19.02	0.1383	no
$H0_{ACC,all}$	6.12	4.85	0.1841	no
$H0_{RC,all}$	0.56	0.50	< 0.001	yes

the higher number of functions and lines of code produced higher values for total cyclomatic complexity of 2008v code compared to 2004v.

The results of hypothesis testing for the stated hypotheses are presented in Table 2. As significance level, 0.05 is chosen.

Concerning $AELOC$, this metric is somewhat increased for changed files, meaning that when files are changed the functions in the files have become somewhat larger. For new files the metric is much lower than for old files, meaning that functions in new files are smaller than in older files. In total, looking at all files, the metric is higher in the new version than in the older version. The differences are statistically different for changed code and new code compared to old code, but not for all code.

Concerning ACC the same type of observation as for $AELOC$ can be made. For changed files the complexity is slightly higher and for new files the complexity is much lower.

For RC there is no significant difference for changed code, but for new code there are significantly less comments. In total there is relatively less comments in the new version compared to the old version.

5 Discussion

The results of the conducted research indicate that in terms of number of files that were changed and updated, source files grouped under the front end component were most affected. The source files grouped under the components library (the src/cl directory) have seen the most of the 2004v source files deleted (file type 0) number-wise. The least amount of changes was seen in the back end component or src/back library. This means that more changes have been made to the "top level" components than the more "lower level" components. There can be many reasons for this, e.g. simply that more changes were needed in these components, but another reason may be that these are nearer to the interest of the new community that was formed during the open source transition process. This is a question that can be investigated in future research.

The metrics pertaining to file type 2 changes indicate that changes made to 2004v source files resulted in significant increase in average cyclomatic complexity ACC and increase in average effective lines of code $AELOC$. A simple answer to why this happened cannot be given, but one possible explanation may be that many times in practice, when maintenance of certain components is done, rather then doing complete re-factoring of the source code affected by the changes, chunks of code deemed too complicated to thoroughly re-factor are surpassed. This way, changes made to source code in terms of the affect on the rest of the system are minimal, but such actions can increase complexity. This too needs further research.

The results of comparison of code quality metrics between all files in 2004v and new files in 2008v show significant and large decrease in ACC and $AELOC$, that is, significant and large increase in quality metrics for code developed by the OSS community. The code quality decrease in metrics smaller than the increase of the changed files, and as a result the code quality metrics for 2008v are higher than those of the 2004v, but this increase in code quality is not significant.

At the same time the number of comments per effective lines of code (RC) has seen significant decrease between the 2004v and 2008v of source code base. Hence, while there was a small improvement in ACC and $AELOC$, the lower number of comments per effective lines of code suggests that code in OSS community was not documented as much as in closed source environment. This is also an input to further research. It cannot at this stage be said if the lower number of comments is an increase or decrease of quality, but it is clear that there has been a change in the way comments are made.

For the companies planning to go open source, this study can provide an example on how the OSS community can have a positive impact on software quality metrics in terms of files that are added to the source code base, but also the negative impact in terms of the files that were changed.

6 Conclusions

The conducted analysis have shown that over half of the changes made to the case source code were made in the front end group of source code components, while the least of the changes were seen in the back end components. The overall code quality metrics, in terms of average cyclomatic complexity and the average effective lines of code per function has increased somewhat for changed code, and decreased rather much for new code. This might be interpreted as an improvement for added code. The number of comment lines per effective lines of code ACC has decrased and there are significantly less comments in newly added code.

The transition of the software was also accompanied by 100% increase in customer base, out of which some 138 customers belong to the Fortune 500 group, and 32% revenue increase reported for the 2008. Hence, the example of Ingres DMBS software migration from proprietary to OSS environment provides one example on how software development can be transitioned from the proprietary environment to OSS community and what kind of impact community can have on the static software quality metrics. To be able to draw some more general conclusions or propose guidelines for improvement of the proprietary to OSS transition process and related software quality metrics, a more analysis of ongoing and completed transition processes should be done.

Based on the presented research it is possible to formulate a number of research questions for further research in the area. In the research it was found that the complexity and size of changed functions increased somewhat. Further research can be carried to understand more about the reasons for this, and to understand if it is an effect that is general for more systems.

It was also found that the length and complexity of functions that were added after the software was transformed to open source were lower. Further, it was found that the amount of comments were lower in code added after the open source transistion. These facts could also be further researched. More research is also needed to determine how these static quality metrics affect dynamic software quality metrics such as performance, reliability etc.

Acknowledgment

The authors would like to express their gratitude to the Ingres Corporation for providing us with a last proprietary version of the software.

References

[Ass09] Matt Assay. *February 2009 Web Server Survey*. `http://news.cnet.com/8301-13505_3-10156188-16.html`. 2009.

[Bon+07] A. Bonaccorsi et al. "Business Firms' Engagement in Community Projects. Empirical Evidence and Further Developments of the Research". In: *First International Workshop on Emerging Trends in FLOSS Research and Development (FLOSS'07: ICSE Workshops 2007)*. 2007.

[FF02] Joseph Feller and Brian Fitzgerald. *Understanding Open Source Software Development*. Addison Wesley, 2002.

[FP98] Norman E Fenton and Shari Lawrence Pfleeger. *Software Metrics: A Rigorous and Practical Approach, Revised*. PWS Publishing Company, ITP International Thomson Publishing Company, 1998.

[HZ07] Ming Gu Hongyu Zhang Xiuzhen Zhang. "Predicting Defective Software Components from Code Complexity Measures". In: *Dependable Computing, IEEE 13th Pacific Rim International Symposium* (2007), pp. 93–96.

[Ing09] IngresWebSite. *Official Web Site of Ingres Corporation*. `http://ingres.com/`. 2009.

[IS02] Apostolos Oikonomou Ioannis Stamelos Lefteris Angelis. "Code Quality Analysis in Open Source Development". In: *Information Systems Journal* 12.1 (2002), pp. 43–60.

[Li+06a] Jingyue Li et al. "A state-of-the-practice survey of off-the-shelf component-based development processes". In: 2006, pp. 16 –28.

[Li+06b] Jingyue Li et al. "An empirical study on decision making in off-the-shelf component-based development". In: *Proceedings - International Conference on Software Engineering*. 2006, pp. 897 –900.

[O'G04] Maureen O'Gara. *CA Exorcises Linux' Hooking Demons*. `http://maureenogara.sys-con.com/node/44941`. 2004.

[RSM08] RSM. *Effective Lines of Code eLOC Metrics for popular Open Source Software Linux Kernel 2.6.17, Firefox, Apache HPPD, MySQL, PHP using RSM*. `http://msquaredtechnologies.com/m2rsm/docs/rsm_metrics_narration.htm`. 2008.

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O'Reilly Media, Inc., 2001.

[RH08] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2008), pp. 131–164.

[Sof08] Coverity Software. *Open Source Report*. `http://scan.coverity.com/report/Coverity_White_Paper-Scan_Open_Source_Report_2008.pdf-`. 2008.

[Som07] Ian Sommeville. "Software Engineering". In: Addison Wesley, 2007.

[VKG06] James D. Herbsleb Vijay K. Gurbani Anite Garvert. “A Case Study of a Corporate Open Source Development Model”. English. In: *Proc. International Conference on Software Engineering (ICSE)*. 2006, pp. 472 –81.

[Woh+00] Claes Wohlin et al. *Experimentation in Software Engineering: An Introduction*. The Kluwer International Series In Software Engineering. Kluwer, 2000.

A Case Study of Open Source Development Practices within a Large Company Setting

Abstract

Open source communities have demonstrated that complex and enterprise grade software can be produced, supported, and maintained by self-organizing groups using primarily electronic form of communication. Due to the inherent nature of open source development, a specific set of open source software development practices has evolved. While there is an ongoing research on the topic of open source development practices and their implementation within a company setting, still little is known about their benefits and challenges. The objective of this research is to understand if and to what degree open source development practices observed within a mature open source community are aligned with development practices within a large company setting. For the purpose of this case study a set of open source development practices that are present in a mature open source community has been defined. Then, development practices of a major hardware and software company were assessed. It is shown that there are many similarities between a mature open source community and a large company setting in regard to software development practices. We also identify practices that exist in open source communities and that are not standard within a company setting, but whose

Alma Oručević-Alagić, Martin Höst

implementation can result in an improved software development efficiency within the company setting.

1 Introduction

In the past two decades Open Source Software (OSS) has emerged as an important player in the field of software production e.g. [Ray01a]. Many companies traditionally operating around closed source/proprietary software have taken interest in different aspects of the OSS. Some companies use open source as another type of off-the-shelf software, while others use it as a part of their software or hardware product and/or even choose to actively participate and contribute code to open source communities (OSCs), see for example Höst and Oručević-Alagić [HOA10b].

Software production is a complex engineering endeavor, and the more people are added to an ongoing software project, the more development efficiency decreases due to need for 'ramp up' time and an increase in communication overheads [Jr.95]. Large OSCs, like the ones under the Apache Foundation [Apa12b] with over 100 OSS projects, have self-organized to produce complex enterprise-grade software. Most of the developers contributing to OSS projects are unpaid, geographically distributed, and yet integrated into highly organized and structured fabric of OSS projects [Ray01a]. Scacchi [Sca10] argues that OSS development is an interesting alternative approach to development of large systems and suggests that further research, especially using empirical examination, is needed in order to better understand OSS development practices (OSDPs). While there exist many OSS projects, very few are active. Research has shown that most of the OSS projects fail, and very few of them succeed in building a sustainable community capable of producing enterprise-grade software [HIC06a]. However, when OSCs are sustained more than a decade, such as Apache or Linux, it is interesting to further examine OSDPs of such communities.

Fogel [Fog05] provides an in-depth analysis of what it takes to build a sustainable open source community, such as the one existing around the Apache Subversion project [Apa12a]. Hence, through study of Fogel [Fog05] and Apache Subversion project [Apa12a], a set of OSDPs has been defined. Then, software development practices (SDPs) of a large international hardware and software company based in Sweden were assessed. The company, which will be referred to as the Case Company, is a relevant case to study since its core software is built around OSS, and many of the team members are active participants of different OSCs. The intention is that the comparison of OSDPs and SDPs of the Case Company will identify similarities and differences, which can later serve as a starting base for further analysis.

The outline of this paper is as follows. In Section 2, the background information on OSDPs is presented. In Section 2, the research approach is further defined,

thus stating what the relevant research questions are, analysis approach and validity concerns. Section 3.2 presents the obtained results, while Section 4 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 5.

2 Background and related work

Studies conducted in HP [MM08], Lucent [GGH06], and Nokia [LRM08a] have investigated possibility of building software products using OSDPs within a confined environment defined by closed organization boundaries. The studies analyzed development of a complex software product across departments [GGH06] using OSDPs and transitioning of an entire development to adopt OSDPs [MM08]. The software produced in such manner is called "inner source" (ISS), "progressive open source" (POS), or "closed open source" (COS).

A case study conducted by Stol et al. [Sto+11] focuses on the challenges of building and integrating software products developed as a shared asset. In this case study the focus is on challenges of developing and integrating software developed as a shared asset within the company setting (ISS), and comparing these challenges with the challenges of integrating an open source software product developed outside of the company. The Stol et al. [Sto+11] research concludes that organizations can benefit in adoption of OSDPs, but that more research in this area is needed to further identify and address the challenges of OSDPs within a company setting. Melian and Mähring conducted a study in HP [MM08] observing the process of progressively transitioning HP's development team to work under OSDPs. Motivation for introduction of POS in HP is a business need to increase development cost efficiency and shorten time to market by making software highly modular and reusable asset. The research has produced a comparative listing of open source and POS development practices. Some of the biggest differences between the two practices lied in the aspects of organizational structure, time and budget to deliver, abundance of available human resources and reward system. They also conclude that implementation of OSDPs within a corporate setting can bring long lasting benefits in terms of development efficiency and code quality, but also state that more research is needed to address differences in reward system, and control and monitoring of individual participants.

A case study conducted by Gurbani and Gavert in Lucent [GGH06] provides another relevant insight into what happens when a software product is developed within a company as a shared asset, and employees from other departments are involved in its development through development process compliant with OSDPs. The lessons learned form this case study are that source code ownership and "many eyeballs" contributing to transparent development process facilitate efficient software development especially if the software product is shared and highly utilized across different departments as this was the case in Lucent [GGH06]. All of the

studies identified importance of having a common set of standard development tools, a single version control system, and standardized change management system.

The Apache Subversion project [Apa12a] is an open source version control system with widespread use in open source and corporate setting. The project was first released as open source in 2000 by CollabNet [Col12]. In 2009 the project became a part of Apache Incubator [Apaa], while in 2010 it became a top level Apache project [Apab]. The Apache Subversion [Apa12a] is used widely by OSC in projects such as Apache Software Foundation [Apa12b], FreeBSD [Fre], SourceForge [Sou], GoogleCode [Goo], and in corporate world according to Forrester [For]. Fogel [Fog05] offers valuable insight on how Subversion [Apa12a] community has been built and sustained over the past twelve years. Besides analyzing the infrastructure needed to host the project tools and, thus, provide a common place for the community's users, testers, and developers to interact, Fogel [Fog05] also elaborates on the importance of building a healthy environment culture, facilitating authority based on meritocracy and communication relying on standardized channels and formats.

The main characteristics of OSDPs were derived from Fogel [Fog05] and thorough study of the actual Apache Subversion project portal [Apa12a]. These major characteristics have been summarized in Table 1 . Hence, the OSDPs' characteristics can be collated under three major aspects:

1. Project web portal infrastructure
2. Communication structure, nature, and norms
3. Participants' roles and rules of engagement within the project

The project infrastructure ensures that there exists a portal that holds information about project such as documentation, source code repository location, binaries and build info, relevant participation information such as mailing lists, issues management, bug info, testing info, and communication archives. The portal should be up to date, intuitive, thus easy to browse and find relevant information. Another, equally important aspect concerns the communication atmosphere and organization. As all communication is in electronic form, the communication atmosphere needs to be comfortable and friendly facilitating team work. Communication channels and norms should be standardized and there should exist an effective way to deal with conflicts. Finally, management based on transparent meritocracy is the key in making sure that the OSC functions properly. Authority is based on merits, and merits are earned and measured through contributions made to the project. It is of utmost importance that the project decision-making structure and roles and rights are clearly defined and followed through consistently.

As a consequence of such setup, OSDPs are characterized by a set of highly standardized communication, management, and development norms. The setup

Table 1: OSDPs' Characteristics

Aspect	Category	Subcategory	I.D.
Infrastructure	Product Info	Features	S1
		Documentation	S2
		FAQ	S3
		News	S4
		Road Map	S5
		Security	S6
	Code Access	Download location	S7
		Binary package	S8
		Release Notes	S9
	Community Guide	Community Overview	S10
		Community Roles	S11
		Coding Conventions	S12
		Commit Conventions	S13
		Building and Testing	S14
		Debugging	S15
		Mailing Lists	S16
		Bugs/Issues	S17
		Releases	S18
Communication	Standardized	Message	S19
		Channel	S20
		Norm	S21
Management	Meritocracy	Role	S22
		Promotion	S23
		Authority	S24

ensures that the participants are aligned on requirements engineering/issues management, bug reporting, source code management, development tools, processes and release management. Hence, the standard software quality levels are shared. Standardized development environment can be conducive to redeployment of developers within or among different projects. Communication transparency and traceability through archives has shown to facilitate cross learning and innovation, and can uncover valuable resources [GGH06].

While the adoption of OSDPs can benefit companies [GGH10], there are also some issues it raises. Some of the issues include development of products across organizational boundaries, especially in the companies where development process is highly hierarchical.

3 Research approach

3.1 Introduction

The study is conducted as a single case study per Runeson et al. [Run+11]. This study is exploratory with the overall objective to investigate alignment of SDPs of the Case Company with the OSDPs as outlined in Table 1. The Case Company is a global market leader in software and hardware production within its field, and its core products are based on an OSS product licensed under GPL. The company has over a thousand employees and, through own and partners' offices, it is present in over 170 world countries. The Case Company is also a significant contributor to different OSCs. Employees in technical and engineering positions at the company have had some level of experience with OSC.

3.2 Research Questions

The following research questions were investigated during the research:

1. How aligned are the SDPs of the Case Company with the OSDPs (Table 1)?
2. What are the underlying causes for the largest differences in OSDPs and the SDPs of the Case Company?
3. Would implementation of the OSDPs with highest degree of misalignment with SDPs within the Case Company benefit or hinder software development efficiency in the long run?

For research question 1 the goal is to identify to what degree the OSDPs are present in the SDPs of the Case Company. Research question 2 analyzes the largest differences between the development practices and their underlying causes. Finally, research question 3 investigates possible short term and long term effects of implementation of OSDPs within the Case Company.

3.3 Data Collection

In this study a qualitative approach has been taken. Figure 1 gives an overview of the research process. Identification of key characteristics of OSDPs present in mature OSCs is based on Fogel [Fog05] and the Apache Subversion web portal [Apa12a]. We refer to an OSC as mature in case the OSC has managed to produce enterprise-grade software widely used by OSCs and the industry, and the community has been sustained for more than five years. In the book Fogel [Fog05] gives detailed analysis on how the Subversion [Apa12a] project evolved over the past 12 years, what were the greatest challenges in establishing OSC as well as how these challenges were resolved. Fogel [Fog05] also analyzes different aspects of the OSC and provides guidelines for their setup. The book study was verified by

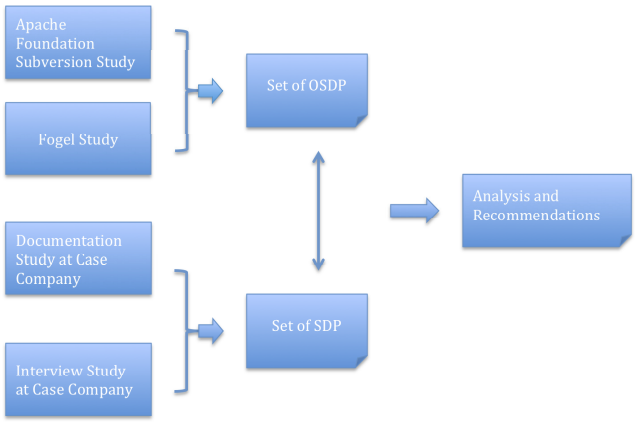

Figure 1: Research Process

analyzing setup of Subversion [Apa12a] project web portal in terms of infrastructure, communication, and roles and responsibilities. The OSDPs' characteristics are summarized in Table 1.

In order to gain understanding of the Case Company's SDPs, on-site documentation was analyzed and interviews were conducted. The first author was granted access to the Case Company, and had conducted a two months long documentation analysis and interview study on-site. In particular, the documentation study assumed analysis and review of documentation hosted on the company's internal portal, such as organizational structure, project development model, info on completed and ongoing projects, training materials, and development standards. The first researcher met four times with an experienced employee from the Case Company to discuss the progress of the documentation analysis and/or get a clarification on any outstanding issue. This is a form of data triangulation according to Runeson et. al [Run+11]. Based on the gathered information from the documentation study, a set of interview questions was created. The interview questions fall into three groups. The first group of questions deals with the profile of interviewee, such as current job role, responsibilities, years of experience within the Case Company and OSC, and rewarding and motivating aspects of the job. The second group of the interview questions analyzes the level of usage of the company's standardized project and documentation portal. It also tries to assess the level of alignment between formal documents on SDPs and the actual-ongoing SDPs within the Case Company. The third group of questions was designed to provide an insight into the Case Company's communication, social and work culture. Interview data was collected through six face-to-face semi-structured interviews with the Case Company employees holding positions of developer, technical lead, IT architect, and project

Table 2: Interview Participants

ID	Position	Years at the Case Company
1	Senior Project Manager	3
2	Code Block Maintainer Code Block Architect Technical Lead	5
3	Technical Lead	2.5
4	Architect	4
5	Developer	5
6	Developer	4

manager. Individual interviews lasted for an hour. All participants had extensive experience and knowledge of the Case Company development practices. All interviews were transcribed and analyzed and coded according to Seaman [Sea99]. The phrases from the answers were initially labeled, and then grouped and merged into categories. Finally, the information gathered from the documentation and the interviews was synthesized and normalized to the form of the Table 1 so the OSDPs versus Case Company's SDPs comparison can be performed more efficiently.

3.4 Analysis procedure

For research questions, the focus is on the information gathered from studies of Fogel [Fog05], Apache Subversion project [Apa12a], Case Company documentation and the interviews. As the core software development of the company is largely based on OSS, members of the development teams have been active members of large and mature open source communities, for over 15 years. All but one interviewee was acquainted with the way OSCs organize and function. In this study the roles and responsibilities of the research and development were of special interest and so more time and effort was put into their analysis while other departments such as sales and marketing were not analyzed.

3.5 Validity

In this section the validity of the research is analyzed with respect to the types of validity threats presented, for example, in [Run+11].

Construct validity: The construct validity is concerned with relationship between the subject of the study and what is measured, in this case the alignment of OSDPs of a mature OSC and SDPs of the Case Company. In order to gain an understanding of the OSDPs of a mature OSC, author used Fogel [Fog05] and verified the results by studying setup and communication trails of the Apache Subversion [Apa12a] project. The Subversion [Apa12a] is an

enterprise-grade software that has been developed by the OSC over a decade. To assess SDPs within the Case Company the first researcher has spent two months within the company, and was seated at one of the company's offices with other company employees. The data on documentation study was collected using a company computer connected to the company's network resources. Thus, prolonged involvement [Run+11] was applied in order to improve the validity of the research. The results of the documentation study were clarified and double-checked with the Case Company employee with senior technical position, i.e. member checking [Run+11]. The results were also reviewed by the second researcher who did not spend time in the company, i.e. peer debriefing [Run+11]. This also reduces the possible bias that the first researcher might have developed with a prolonged involvement. It also means that research triangulation was applied which also increases validity of the research. There exists possibility that six interviewees was not representative sample, but the chances for this are very small.

Internal validity: The internal validity is concerned with causal relations. Since the nature of this study is to compare and analyze development practices, the causal relations do not affect the results of the study.

External validity: The external validity is related to the ability to generalize the results of the this study. The OSDPs as defined in the paper can be relevant for future analysis. The Case Company studied is large software and hardware company, a world leader in its field, where the main products are built around OSS. The competition in the market is typical. Hence, the findings of this research might be relevant to other large software companies that consider implementation of OSDPs internally. It provides a framework of characteristics present in OSDPs and an insight on benefits and challenges of OSDPs internal implementation.

Reliability: The reliability aspect of validity is concerned with the aspect of data and analysis dependence of the underlying research on the researchers. The study was conducted as a structured case study, the analysis and the interviews were done in the structured way, and the interview data was coded according to appropriate methodology thus reducing the risk of data being dependent on the researchers.

4 Results

4.1 Research Question 1: Alignment of OSDPs and SDPs of the Case Company

To answer research question 1, the results of the documentation assessment and the interviews are grouped under the three categories as outlined in Table 1.

Infrastructure

The analysis of the company's internal portal and documentation has uncovered the existence of infrastructure compliant with OSDPs as outlined in Table 1.

The web portal contains documents with information on organization structure, administrative information, roles and responsibilities, information on development processes, methods, standards, past, and ongoing project related information, code repository, use-net groups, and training manuals. Development processes and the methodology is well defined, with a project management process which is best categorized as a set of sequential steps (also called tollgates), which need to be completed before the next step can be taken. The coding standards are clearly spelled out in the documentation. There exists ongoing project documentation mostly with information on project management plans, allocated resources, assigned tasks, and task completion.

The interview study conducted shows that development team members are aware of the existence of the portal and the information that can be found on it. Even though the internet portal infrastructure is present, only two interviewees indicated that they use portal in their daily work, the architect and the senior project manager. These two interviewees use it for the purpose of updating project management plans or technical documents. Developers, code block architects, and technical leads indicated that they use portal very little in daily project related tasks. They also agree that the documents on the portal were not well organized, were hard to search, and that much of the documentation they were interested in was out of date. In case there were multiple projects related to a product, each project had their own version of the documentation, which in itself was outdated.

Communication

The internal portal also hosts infrastructure necessary to carry out discussions on various topics and create searchable archives.

The majority of the interviewees agreed that a great majority of inefficiencies and issues they encounter in their daily work are related to inadequate communication. Most thought that better communication would lead to more efficiency at work. They expressed that usage of electronic communication in a standardized form would be desirable, especially if it would create searchable archives which could later on be referenced for problem solving purposes, similarly to how they would search the internet to understand why programs produce certain error codes and how such issues could be resolved. On the other hand, the majority of the interviewees agreed that it is much more time efficient and easier to 'go and talk' to a person about a problem, recognizing that in this way no written trail on the problem would be left, and thus, no one could refer to it in the future. While there exist non-standardized means to communicate electronically, some interviewees said that the majority of developers refrain from using it, partly due to past experi-

ence, where questions and issues brought up through electronic discussions were not addressed in a manner that would facilitate such discussion.

Management

Interviewees coming from developer, code block architect, code block maintainer, and architect roles had varying levels of involvement with open source communities. In general, the more advanced technical role the interviewee held, the more involvement with the OSC he/she possessed. Except for a project manager, all interviews were users of OSS, and four have contributed bug reports and/or code to and OSS project. The majority of the contributors did this as a part of their work, while some worked on non work related OSS projects at their free time.

The organizational structure and roles and responsibilities within the R&D resemble roles which can be found in the OSC. Hence, besides developers (code contributors), there are technical leads, code block maintainers, code block architects, and architects. The code block architects and code block maintainers can thus be seen as fulfilling the role of 'gate keepers' in the open source community. Similarly to the motivation of OSC members to participate in OSS projects according to Raymond [Ray01a] , the interviewees noted that what motivates them to excel in the job is getting "a kick" from solving a technical problem, ability to create something that works and that is self improving or self sustainable.

The majority of developers were very positive toward the idea of being able to select tasks they would work on from a pool of tasks in the similar manner as this is done in OSC. However, interviewees that were in manager positions indicated that this might not be feasible as then much time would need to be spent on managing conflicts for those developers that could not choose tasks or that were stuck with less interesting tasks.

All interviewees agreed that task deadlines are needed, but sometimes too tight deadlines tend to reflect on quality, as there exists a tendency to put in as much functionality as possible, without properly testing it.

Five of the interviewees thought that the number of formal meetings held was excessive. They expressed that if more time was put in planning of the meeting and appropriate selection of the attendees, the meetings might be less frequent and more efficient. Interviewees at more advanced technical position believed that there was a tendency to involve them into projects too early or too late. This adversely affects efficiency on individual and project level.

Code block architects and code block maintainers noted that in practice their roles overlap with the role of technical lead. Such overlapping roles on the project are conflicting, as technical lead is perceived to be more of a project driver, while code block architects and maintainers are considered to be expert of a product or a part of it with a sole role of making sure that underlying product development is in line with overall architecture.

4.2 Analysis for research question 1: The alignment of OSDPs and SDPs of the Case Company

Hence, based on the carried our research, there is evidence which supports presence of an infrastructure setup, and a role and process organization which is compliant with the mature OSC. However, based on the conducted interview study, the actual day to day business seems to be in a lesser degree aligned with the mature OSC. Thus, even though there exists a portal with information, it is not used much. Developers and, in general, employees in more technical roles find the portal hard to search, and project documentation outdated. Individuals in more management roles tend to use the portal on daily bases and update it with info on a project progress. While there exists communication infrastructure, the usenet groups are very rarely used. Standardized group discussions are the core of communication that enables an OSC to develop OSS products. It also creates searchable archives on past project discussions, and thus can be a very useful source of information when individuals who have worked on a project leave the company. Any support issues and error messages can be stored in such archives, so that in the future similar or same types of issues could be resolved faster.

The existence of archives would also indirectly show who the experts on particular issues are, so that roles of area experts are filled appropriately. Lack of such archives means lower transparency of contributors within a project which hinders the ability of managers to organize resources in meritocratic manner. While there exist area experts, and groups of experts, the interviewees said that the roles are many times overlapping, adding to higher job stress and inability to do the job of 'expert' properly.

Hence, while in the Case Company the three aspects of OSDP are implemented to varying levels of degree, the actual level of alignment between the SDPs of the Case Company and OSDPs has some space for improvement.

4.3 Research Question 2: The greatest misalignment in OSDPs and SDPs of the Case Company

Through the analysis of the research question one, it has become evident that the Case Company has a dedicated portal to host information as described in Table 1, and marked with I.D. S1-S18. However, the provided portal content in many instances is not complete, up-to-date, searchable or formatted so it can be utilized to its true potential. Hence, the portal infrastructure facilitating OSDPs is present, but the content is not relevant and used by software development resources of the Case Company in the degree and manner that it is relevant and used by the participants of the OSCs.

The communication architecture with characteristics S19-S21 is formally not present. Usenet groups are sporadically used. Electronic transparent discussions are not formally enforced. Communication archives as such do not exist. Histor-

ical information on completed and ongoing projects can serve to some extent as searchable communication/project history archive, but in much lesser sense than the communication archive is intended to provide such information in the OSC.

4.4 Research Question 3: Misaligned practices and their implementation within the Case Company

The following SDPs of the Case Company showed the highest level of misalignment with the OSDPs. Earlier research has shown that these OSDPs can improve efficiency in the company setting.

1. Well organized, easily searchable infrastructure portal holding relevant and up-to-date information for R&D team.
2. Up-to-date documentation on product.
3. Searchable communication archives and issues database.
4. Communication standards and norms in electronic form.
5. Communication atmosphere which facilitates group discussions.
6. Non-overlapping contributors' roles.
7. Higher degree of transparency for an ongoing project, from its inception to completion.

5 Discussion

The fabric of OSCs is defined by the information hosted by the web portal and the OSCs' "code of conduct". The portal is the only interface OSC member has with the project worked on, and thus it is the sole source of information about the project, its purpose, development, issues, and future direction. As such the role of the portal or the common OSC's playground is to attract, educate, and retain contributing participants with the goal of further improving its OSS product.

The Case Company has recognized benefits of using and further improving the software developed by OSC. It has also replicated many of the roles present in OSC, such as the role of code block architects, technical leads, head architects, code maintainers. The Case Company has also implemented the portal and formally set up the usenet groups. Development resources are highly aligned with OSDPs in respect to common understanding of technical issues and value of standardized practices in the respect to design, coding, testing, and development stages. They also recognize value of code ownership, and existence of technical roles found in OSC, sure as code gatekeepers, code block architects and technical committees.

The OSCs attract new participants through the project portal. Since there exists no formal individual training process for a new OSC participant, the portal, that is, the information hosted on it, plays this role. Thus the portal of a mature OSC has complete, unambiguous, and up-to-date documentation on project features, direction, technical aspects and OSDPs. It is expected that the new OSC participant will acquire all necessary information to become a member of such community through the portal resources. On the other hand, in the Case Company much of this educational aspect of portal is shifted to human resources, mostly their peers. Four of the six participants from the interview study have indicated that updating project documentation is one of the lower priorities since normally there exist tight project deadlines and many times time planned for such activity is spent on issues with higher priority. While in the short run not completing project documentation may speed up the completion of the project, in the long run it creates incomplete and out of date project documentation. The four interviewees also indicated that project documentation in many instances is ambiguous if, for example, there exist multiple development projects around one software product and the projects do not sync-up the product documents. This way, the new employees working on some future projects around the software product, and in need of such information will have to spend time with resources which were involved in the project to understand particular product features, design decisions, architecture, etc. If the resource leaves the company, then an additional amount of time will need to be spent analyzing perhaps the source code to obtain the relevant information.

Some of the main misalignments between the SDPs and OSDPs are seen in the area of communication and management aspects of the project. While majority of developers would like to have searchable communication archives, a starting place where one could go to find out if there exists more information about an investigated issue, they perceive that communicating electronically instead of "face-to-face" is less efficient. The interviewees also indicated reluctance to part take in open electronic discussions and such discussion are not formally encouraged or enforced to any degree. Face-to-face communication reduces the amount of time one needs to spend searching through archives and can also "on-demand" ask for further clarification of an issue. On the other hand, 'face-to-face' talk can be less efficient in case the resource one needs to talk to is not currently available. The resource might need to repeat the same sort of information or give similar clarifications over and over again. The closed, non-transparent nature of such communication might result in exclusion of relevant decision makers. Two of the interviewees indicated that meetings concerning an issue are called prematurely or too late, and many times they do not involve appropriate resources. Majority of the interviewees have expressed that coordination and communication takes up large amounts of their time. Hence, 'face-to-face' communication can be more efficient in some cases in short-run, but in the long-run establishing communication norms, appropriate mailing lists or forums, and archiving discussions might improve efficiency.

The Apache Subversion project states that Subversion uses a compromise between time-driven and feature driven release planning. The community participants believe that a new feature needs time to mature, and thus they do not want to force technical discussions to premature consensus. In a market oriented organization it can be difficult to afford time for a technical discussion to slowly mature. Such environment is functioning within time constraints formed by customers' needs and market competition. Normally developers in a closed company setting are separated from the end users and through intermediaries learn of the issues or newly requested features. Hence, the aspect of truly transparent communication not only within the company among peers, teams or departments, but also towards the end users can not be easily established in the closed company setting. As such, OSDPs and SDPs can not be fully aligned.

Since there exists an interface between the end user and development team, the interface comprised of project sponsor, project manager, project owner, a lot of time is spent on coordination and communication of the relevant project players. This highly time intensive activities which are crucial in creating the software product per end user needs, would under OSDPs happen transparently through issue/bug reporting fora. As mentioned earlier, OSCs are not under time/market constraint to push out certain features, rather they can afford time and let a feature discussion mature.

6 Conclusions

The assessment of the company has shown OSDPs to be implemented to a higher degree in a form of infrastructure, and less in a form of communication and management practices. Hence, there exist technical roles modeled around software product, such as head architect, code block maintainers, code block architects. There also exist standardized development practices and processes facilitating cross project work. The Case Company portal is created with purpose of mimicking an OSC portal, but in practice the contents of the portal are not well organized, complete, or up-to-date, and in many instances are ambiguous and hard to search.

The communication aspect is relaxed, where unlike the OSC whose rules of communication conduct is in strict standardized, archived, and transparent form, the communication is face-to-face, and important discussions are done within dedicated groups perceived as appropriate decision makers. Management of the resources is in line with project planning and existence of interface between the end user and development.

Some of the differences seen can be justified by time to market constraints, such as higher degree of management involvement and task assignment. However, aspect of communication is largely misaligned with OSDPs due to perceived effectiveness of "go and talk" to the person. Effective communication process can,

in the long run, change or evolve roles currently set up around projects, as it can bring clarity to which positions are truly needed.

Further research is needed in form of similar case study research or experiments to test applicability of OSDPs within the closed company setting and thus to clarify to what extent further implementation of OSDPs within the closed settings can benefit the companies.

An interesting research topic for the future would be to conduct an experiment in a close company setting mimicking requirement gathering of OSDPs and letting internal resources participate in online discussions and selection of features and tasks. This in-house OSC could then be observed and analyzed in order to gain a better understanding of challenges and benefits of such software development process.

Acknowledgment

This work was funded by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se)

References

[Apaa] *Apache Incubtor*. http://incubator.apache.org/. 2012.

[Apab] *Apache Top Level Projects*. http://projects.apache.org/. 2012.

[Apa12a] Apache. *Apache Subversion Open Source Project*. http://subversion.apache.org/. 2012.

[Apa12b] Apache. *The Apache Software Foundation*. http://www.apache.org/foundation/. 2012.

[Col12] CollabNet. *CollabNet Community*. http://www.collab.net/. 2012.

[Fog05] Karl Fogel. *Producing open source software - how to run a successful free software project*. O'Reilly, 2005, pp. I–XX, 1–279.

[Fre] *FreeBSD Open Source Project*. http://www.freebsd.org/. 2012.

[Goo] *Google Code*. http://code.google.com/. 2012.

[GGH06] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. "A case study of a corporate open source development model". In: *ICSE*. 2006.

[GGH10] Vijay K. Gurbani, Anita Garvert, and James D. Herbsleb. “Managing a corporate open source software asset”. In: *Commun. ACM* 53.2 (2010), pp. 155–159.

[HOA10b] Martin Höst and Alma Oručević-Alagić. “A Systematic Review of Research on Open Source Software in Commercial Software Product Development”. In: 2010.

[HIC06a] James Howison, Keisuke Inoue, and Kevin Crowston. “Social dynamics of free and open source team communications”. In: 2006, pp. 319–330.

[Jr.95] Frederick P. Brooks Jr. “The Mythical Man-Month: After 20 Years”. In: *IEEE Software* 12.5 (1995), pp. 57–60.

[LRM08a] Juho Lindman, Matti Rossi, and Pentti Marttiin. “Applying Open Source Development Practices Inside a Company”. In: *OSS*. 2008, pp. 381–387.

[MM08] Catharina Melian and Magnus Mähring. “Lost and Gained in Translation: Adoption of Open Source Software Development at Hewlett-Packard”. In: *OSS*. 2008, pp. 93–104.

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O’Reilly Media, Inc., 2001.

[Run+11] Per Runeson et al. *Case Study Research in Software Engineering*. Wiley, 2011.

[Sca10] Walt Scacchi. “The future of research in free/open source software development”. In: *Future of Software Engineering Research*. 2010, pp. 315–320.

[Sea99] Carolyn B. Seaman. “Qualitative Methods in Empirical Studies of Software Engineering”. In: *IEEE Transactions on Software Engineering* 25.4 (1999), pp. 557–572.

[Sou] *Source Forge Project*. `http://sourceforge.net/`. 2012.

[Sto+11] Klaas-Jan Stol et al. “A comparative study of challenges in integrating Open Source Software and Inner Source Software”. In: *Information & Software Technology* 53.12 (2011), pp. 1319–1336.

[For] *Subversion Sole Leader for Standalone SCM in SCM Forrester Wave report; 2007*. `http://blogs.collab.net/subversion/2007/06/subversion_sole/`. 2012.

NETWORK ANALYSIS OF A LARGE SCALE OPEN SOURCE PROJECT

Abstract

Industry involvement in open source software development has become a popular practice among companies which, e.g., share software development costs with other community participants or implement an open source based business model. An increased understanding of the underlying social structure and influences within an open source community, especially in a case where the community participants are composed of competing industry members, can be viewed as an important component of company's business strategy planning and management. One way to understand the social structure of an open source community is by applying social network analysis to source code repositories. We use methodology from social network analysis to study Android, an operating system for mobile devices.

The aim of this study is to understand how large and in many cases competing companies collaborate on a large scale company sponsored open source project. We propose a new approach for studying committers' networks as weighted digraph networks. To conduct the case study, change log records of all files bundled under the Android open source project stack were extracted and studied in the context of committers' networks.

The results obtained show that Android project development is highly influenced and led by development effort of Google.

Alma Oručević-Alagić, Martin Höst

This case study shows how a large, company sponsored, and industry backed open source project, i.e. open source project with the majority of community members affiliated with the industry, is structured. In particular, it shows that the involvement of an entire industry eco system within a company sponsored open source project does not imply more equal distribution of the participating community members' influences in terms of committers' social networks. This setup of an open source community by itself does not imply any particular, either positive or negative, connotations. Consequently, the results of the study should be interpreted on a case bases, within a context of a company's strategy to participate or base products around a company sponsored open source product.

1 Introduction

Open source software (OSS) has been growing in importance and affecting the way companies develop their products and services [HOA11b], plan their business strategy and compete [Ray01a].

Many companies have already recognized that implementation of software product lines can facilitate software development in an assembly line like manner. A study by Linden et al. [LLC08] shows that software product lines can decrease software production cost. In a similar manner, an OSS product can be reused across different companies, e.g., Android phones produced by Samsung, Sony Mobile, HTC, etc. Another study by Linden et al. [LLM09] refers to software reused across an industry as commodity software. The study argues that a part of a software product over time loses commercial value, and thus becomes a good candidate for intra-industry or open source development. In this way, development costs of commodity software become shared among the industry members, enabling companies to focus their resources on development of commercial or differentiating parts of their software products.

There have been many studies conducted on open source projects by analyzing source code change logs and mailing list archives in order to understand the underlying structure and behavior of the community. The studies focused either on some individual open source projects [LF+06], or on an entire portal hosting tens of thousands of open source projects [HIC06b]. Different methodologies have been applied to analyze the obtained data including social network analysis. As open source communities are composed of geographically distributed participants that contribute on volunteer or paid basis (e.g. company participating in OSP), social network analysis has shown to be an effective methodology to analyze participants' affiliation networks, identify the most influential participants, and uncover cliques. The Android was initially developed as proprietary software by the Android corporation. In 2005, Google Inc. bought the Android [And05] and open sourced the operating system in 2007 [And07] . At the same time, Google also founded an open handset alliance [And07]. The open handset alliance is a con-

sortium of over eighty globally leading companies in market segments of mobile operators, handset manufacturers, semiconductors, and software and commercialization companies. The companies contribute to the development of the Android and deliver devices and services built around the Android operating system. Companies like Vodafone, Sprint, T-Mobile, Acer, HTC, Samsung, Sony Mobile, Arm, Intel, ST Ericsson, eBay, Accenture, are some of the members of the alliance.

Besides the core components open sourced by Google in 2007, the Android also includes over 150 other open source projects, the majority of which were in existence before the Android project. One such project that is included under the Android bundle is WebKit, which was initially developed as proprietary software and later open sourced by Apple. Hence, another interesting aspect of the study is assessment of cross collaboration among participants of various open source projects bundled under the Android stack.

The Android source code committer's network is analyzed through application of social network analysis. The committers are grouped based on affiliations, i.e., all contributors affiliated with a company or an organization are viewed as the same contributor, e.g., committers with a google.com email suffix are viewed as Google company committer. The affiliation data such as authorship information and contribution date and time are extracted from the source code repository logs.

The aim of this study is relevant given the importance of the Android in the mobile device industry and diversity and type of participants within the Android open source project. For any company that plans to either sponsor, lead, contribute to, or build products based on an OSS product, understanding of project's underlying community structure and behavior is an important factor in the company's strategic positioning. This paper shows one way to assess the structure of an open source community by applying social network analysis. Given that Android includes external open source projects, i.e. the projects not developed or open sourced by Google, it is also important to understand how the underlying community structure of the external projects can be affected when the projects are used by another company sponsored open source project with global industry support.

The outline of this paper is as follows. In Section 2, the background information on the case software and related research studies is presented. In Section 2, the research method is further defined. Section 3.2 presents the obtained results, while Section 4 discusses and analyses the obtained results in some more detail. Finally, conclusions are drawn in Section 5.

2 Research approach

2.1 Introduction

The study is conducted as a case study [RH08]. The investigated case is committers' network structure of Android OSP. The study is exploratory with the overall

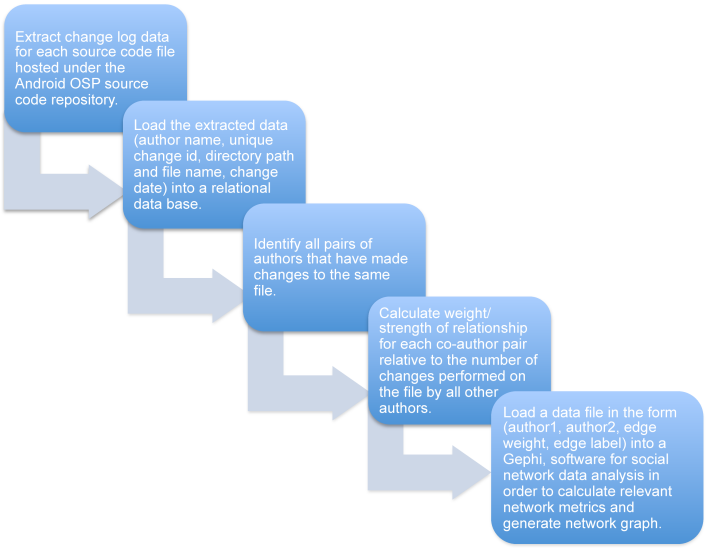

Figure 1: Analysis Process

objective to understand how the community participants collaborate in development of the software through the Android OSS process. The data of the study was collected according to the process presented in the Figure 1. Thus, change log data was extracted for each file within the Android repository, and loaded into a database to simplify data manipulation process which identified all pairs of authors that modified the same file. For each identified co-authorship par, weight and direction of the relationship was calculated. Finally, this data was loaded into the Gephi[Gep13], software for network visualization and analysis.

In this study a quantitive approach has been taken. A study by Luis et al. [LF+06] has shown how social network analysis methodologies can be used to study OSS projects in order to characterize the projects' evolution over time as well as the projects' structure. Affiliation networks are a special type of social network where two distinct sets of actors are related, e.g., a committer network relates a set of committers to a set of changed source code modules. Hence, there exists a link between two committers when they have changed a same module. An actor or a network node is referred to as a vertex and the links between the vertices are called edges as shown in Figure 2.

In this study we propose an approach for studying committers' networks. In Luis et al. Luis [LF+06] the proposed methodology establishes links between the committers, where the weight of the link or the edge is calculated as being the number of commits performed by committers to all common modules, i.e. the degree of relationship. The definition of the common module differs between projects,

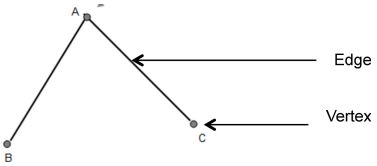

Figure 2: A Three Actor Network

but usually corresponds to the top level directories of a source code repository.

According to Borgatti and Halgin [BH11] an important factor to consider when studying strength of the co-affiliation among an event's participants is the actual size of the event. The research suggests that one of the ways to normalize the strength of a co-affiliation between event participants is to weight participation relative to the size of event. In the studied context, the size of the event is the total number of changes made to same file. Then, the strength of co-affiliation among participants relative to the size of event can be expressed as the number of file changes performed by each participant relative to the total number of the changes performed on the file by all participants. For example, if two companies, A and B, make changes to same file, where company A makes only a few changes while company B makes a majority of the changes, then the influence of A over B is much smaller than the influence of B over A relative to the size of the event. A study by Hangal et.al [Han+M] also examines asymmetric influences of nodes through a friendship example and infers weights on friendship relationship. Hence, we propose a new approach to study committers networks as weighted digraphs as shown in Figure 3. The figure shows weights of the edges for committers associated with companies A, B, and C who have changed the same source file 5, 10, and 15 times, respectively. The edge weight is calculated as the number of the committers' changes on a file relative to the total number of changes for the file, which in this case is 30. Thus, the committer A infers a weighted influence of 1/6, B of 1/3, and C of 1/2 to the files's co-committers.

In Riitta Toivonen [Toi+07] argues the importance of strengths of edge ties when modeling social structure and dynamics of social networks. We argue that inferring the edge weight relative to the size of the event provides a more accurate social network structure from the one suggested by Luis et al. Luis [LF+06] which does not take into account relative size of event. For example, if only a degree of relationship is considered in the above example for the committers A, B, and C for, e.g., the total number of files they changed together, then the edge weights between the three committers would be the same. This would mean that the strength of co-affiliation between A, B, and C is the same relative to the source file change event, which is clearly not the case. While this is a simple and trivial example, in a context of a large network, with many committers, where, e.g, a subgroup of committers performs a large number of changes, computing edge weights relative

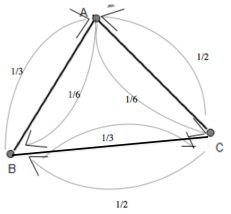

Figure 3: A Weighted Three Actor Network for Modification of One Source Code File

to the number of all changes performed on a file is important in order to accurately assess relationship strength. This is more so as the data on committers, corresponding edges, and their weights are building elements of a network structure, based on which other network metrics are derived.

In this study, the weight of the edge between two participants is calculated on a file level. Affiliation networks link actors into a social network by virtue of participants attending a specific event. In the context of committer network analysis we define the event as performing modifications on a specific source code file. Hence, for a set of actors $V = \{v_1, v_2, ..., v_k\}$ and events $U = \{u_1, u_2, ..., u_m\}$ we define a weight W of an edge between an actor v_i and all other actors that participate in the event u_t as:

$W(v_i, u_t) = \frac{X(v_i, u_t)}{\sum_{c=1}^{k} X(v_c, u_t)}$

where $X(v_i, u_t)$ denotes the number of times an actor/committer v_i made changes to the file, i.e., participated in the event u_t.

This means that the weight of the edge $W(v_i, v_j)$ for all events v_i and v_j attended together equals:

$W(v_i, v_j) = \sum_{t=1}^{m} W(v_i, v_j, u_t)$

In order to obtain committer data on the source code changes, Android project source code repository was downloaded in November 2012 from the Android project web site [Inc13]. Change log records, with information on authors and change dates for all Android source code files were extracted and loaded into a database. The social network data on network nodes/committers, edges, and associated edge weights and labels was analyzed using Gephi software for social network analysis [Gep13]. The labels correspond to the main subdirectories under the Android source code tree, as displayed in table 1. The Gephi software was used to calculate relevant social network metrics which are discussed in more detail in the Section 2.4 as well as to generate a visual representation of the committers' networks. Besides analyzing committer network for the entire repository, we also analyze two additional distinct committer sub-networks. The tree committers' subnetworks are constructed:

1. External committers network which includes committers that changed files

located under the external top subdirectory.

2. Core committers network which includes committers that changed files located under all top subdirectories excluding the external subdirectory.

3. The entire committers network which includes committers that changed files located under both, the core and the external subdirectories.

Since the external subdirectory contains source files for over 150 other open source projects, we believe that studying this diverse community separately from the core Android community can provide some additional insight. Committers that participate in the external open source projects do not necessarily use the Android OSS or participate directly in its community process. Distinguishing between the core and external committers can also provide an additional insight into committers that work under the Android OSS project. Finally, a combined network of all, the external and the core committers is studied.

2.2 Research questions

The following research questions were investigated during the research:

1. What are the characteristics of the committers' networks for each set of the Android project source files: the core, the external, and combined core and external?

2. How can a company utilize network analysis to study the Android development community?

For research question 1, the focus is an assessment of the three distinct network structures, the core components committer network structure, the external committer network structure, and the combined core and the external committer network structure. Metrics on network influence, clustering, centrality, existence of sub-communities, and network density are presented and discussed.

For research question 2, we analyze results of research question 1 from the perspective of a company planning to develop software through Android or similar OSP.

2.3 Investigated software

A program that parses trough all source files located under the Android OSP subdirectories (table 1) was created and run in order to collect information on all the changes made to all the source files in terms of authors and change dates. The extracted data was loaded into a relational database in order to perform a thorough data validation and provide flexible way to create different file input formats for the Gephi [Gep13] social network analysis software. All authors were grouped

Table 1: Android Subprojects or Modules

Top Level Subdirectory	Description
abi	Features
bionic	The C-runtime library for Android.
bootable	Boot and startup related code.
build	Utilities and scripts for building system implementation.
cts	Android compatibility testing framework.
dalvik	Android virtual machine.
development	Source code for SDK and NDK, and emulator.
device	Product specific code for different vendor devices.
docs	Tutorials, references, and miscellaneous information.
external	External open source projects (WebKit, SQLite, etc).
frameworks	Key Android framework library (JNI, services, phone and telephony components, etc)
gdk	Compiler infrastructure for the NDK based on LLVM
hardware	Libraries for basic hardware support.
libcore	The Harmony Java Virtual Machine used by Dalvik.
libnativehelper	Native development helper library.
ndk	Native Development Kit.
packages	Source code for default Android applications (e.g. calendar, contacts,etc).
pdk	Platform development kit provided to chipset vendors and OEMs before new platform is released.
prebuilts	Files distributed in binary form.
sdk	Android Software Development Kit.
system	Core Linux system libraries.
tools	Development Tools.

based on a company affiliation. The affiliation is determined based on committer's e-mail domain suffix. In case email data was not provided, authors individual names are used and no company affiliation is implied. All of the contributions made by the author named "Initial Android Open Source Project Contribution" was excluded from the analysis, as these contributions were not developed under the Android OSS community process, but internally by Google before the project was initially open sourced. The Gephi data records are of the form "source, target, edge weight, edge label". If we consider the earlier example depicted in Figure 3, a sample record would look like "A, B, 1/6, the changed source file's top subdirectory".

2.4 Metrics

The following metrics were measured for the three committer networks:

- Weighted average in-degree $WAID$ and weighted average out-degree $WAOD$ of a vertex.
- Betweenness centrality BC, closeness centrality CC, and eigenvector centrality EVC of a vertex.
- Average Clustering Coefficient ACC of a vertex.
- Modularity MC of a network.
- Number of MCN of a network.
- Graph density GD of a network.

textitWeighted degree of a vertex denotes degree of relationship of the vertex with its direct neighborhood. It is calculated as the sum of weights of all edges connected to the vertex. Since the analyzed network is weighted digraph, there exist two types of edges; the edges originating from a vertex, or the out degree ($WAOD$), and the edges pointing to a vertex, or the in-degree ($WAID$). Hence, the weighted average in-degree of a vertex denotes degree of relationship of the vertex to its direct neighborhood for the edges pointing to the vertex. By the same analogy, the weighted average out degree of a vertex denotes degree of relationship of the vertex to its direct neighborhood for the edges originating from that vertex. In the context of committer network analysis, the out degree can be interpreted as the measure of collaboration strength or influence of the committer on committers in its direct neighbourhood. The in degree can be interpreted as the measure strength of influence of committers in direct neighborhood of the committer on the committer.

textitBetweenness centrality index (BC) is the number of shortest paths that traverse through a vertex and it can be interpreted as a measure of importance of the

vertex in a graph. The higher betweenness centrality index of a vertex, the more important the vertex is. In the context of this study, the betweenness centrality index indicates the number shortest distance paths between any two committers which traverse through a committer.

textitCloseness centrality CC indicates how close on average a vertex is to all other vertices. A high value of the distance centrality index identifies vertices that are well related.

textitEigenvector centrality EVC metric measures the influence of a vertex on a network by assigning scores to all vertices in the network. The scores are assigned so that an edge to a higher scoring vertices is valued more than the same edge to a lower scoring vertex. The eigenvector represents the most accurate metric for influence of a vertex on other vertexes in the network. Gephi uses an algorithm by Brandes [Bra01] to calculate the centrality network measures for weighted graphs.

textitAverage clustering coefficient ACC of a vertex shows the tendency of the network to form cliques or isolated groups. The average clustering coefficient is calculated based on sum of individual clustering coefficients. The individual clustering coefficient is calculated as the number of edges from a vertex to its direct neighborhood relative to the number of links that could exist between them [WS98].

textitModularity of a network MC identifies the sub-communities within the network with densely connected vertices. The value of modularity is calculated as a difference in fraction of edges that fall into the sub-communities and a fraction of edges that could be found in the sub-communities if the edges were distributed at random per Blondel et. al[Blo+08]. In the context of the committer network study, the modularity class is used to identify committer sub-networks with higher degree of collaboration. The modularity value falls between the values of -1/2 and 1, where negative number indicates that a random distribution the edges is more likely to form sub-communities than the actually identified sub-communities. The modularity class number MCN indicates the number of identified sub-communities for a given modularity class MC.

textitGraph density index GD measures how close the network is to being complete, i.e., that there exist edges between all the vertexes in the network. A value of 1 for the graph density index indicates a fully complete or connected network.

2.5 Analysis procedure

Analysis with respect to research question 1 was conducted by calculating the $WAID$, $WAOD$, BC, CC, EVC, ACC, MC, GD on the core, external, and combined core and external Android OSP source code tree. For research question 2, the presented network structure data in question 1 is analyzed from a business/-company perspective.

2.6 Validity

In this section the validity of the research is analyzed with respect to the types of validity threats presented, for example, in [RH08].

Construct validity: The construct validity is related to the relationship between the concepts and theories behind the experiment and what is measured and affected. The subset of metrics from the network theory used in this research has been accepted and validated in other studies within the filed of OSP repositories and mailing archive studies. This means that the risk of using metrics that do not represent the concept of social network structure is lowered.

Conclusion validity: The conclusion validity is concerned with the possibility to draw correct conclusions regarding the relationship between treatments and the outcome of an experiment. The interpretation of the metrics is grounded in the widely accepted network theory and the field of social network analysis.

Internal validity: The internal validity is concerned with factors that may affect the dependent variables without the researcher's knowledge. The data extracted from the repositories was examined and validated manually through sampling. The approach used in constructing committers network is grounded on network theory concepts applied in other disciplines.

External validity: The external validity is related to the ability to generalize the results of the experiments. The studied software is relevant example of a successful industry led OSP as the project includes leading global companies from the mobile eco system.

3 Results

3.1 Research question 1: An assessment of the three distinct network structures, the core components committers' network structure, the external OSPs committers network structure, and the combined committer's and external network structure.

The core committers' network has a total of 250 vertices and 3606 edges, which in this case means that committers have 250 distinct affiliations and there are 1803 distinct committer co-authorship pairs. Since the network is modeled as a weighted digraph, the edges are bi-directional. The external committers' network has 329 vertices and 11196 edges, while the combined core and external committers' network has 513 vertices and 14484 edges.

Table 2 shows ACC, MC, MCN, and GD for the three studied committer network structures. The average clustering coefficients for the three networks show high tendency of the networks to form cliques.

Table 2: Summary of the committers' networks measures

Metric	Core	External	Core and External
ACC	0,782	0.791	0,799
MC	0,0009	0,356	0,43
MCN	4	6	7
GD	0,058	0,104	0,055

The identified number of closely related sub-communities MCN for the core committer network is 4. However, the MC value of 0,0009 indicates that a probability of such sub-communities occurring at random is very high. Hence, the identified potential sub-communities for the core committer network should be disregarded since their existence is not statistically significant. The number of sub-communities identified within the external committer network is 6 with the MC value of 0,356 indicating that existence of the 6 subnetworks is statistically significant. The number of identified sub communities for the combined, external and internal committers' networks is 7, with the MC value of 0,43 indicating that the existence of the sub-communities is statistically relevant.

The graph density metric GD for the core, external, and combined core and external committer networks is 0,058, 0,104, and 0,055, respectively. The value of 1 for GD indicates that all the components within the network are highly connected. Hence, all three types of the committers' network showing low graph density values indicate that the committers' networks are weakly connected. The high clustering coefficient shows that even though many edges between the committers are absent, committers in a direct neighborhood of a committer are well linked.

Figure 4 displays the weighted average in-degree and out-degree for the top 16 committers in core committers' network. As noted earlier, the out-degree in the weighted directed network indicates influence of a vertex over other vertices in its direct neighborhood. Hence, committers affiliated with Google have the highest $WAOD$ value. The value is also more than two times higher than the $WAOD$ value for the second most influential group of committers, that is the group of committers are affiliated with Android OSP community. The $WAOD$ metric decreases tenfold for the third highest rated committer group affiliated with the Gmail.com address as compared to the Google. The weighted average in-degree $WAID$ metric for Google representing the influence of all other committers in Google's direct neighborhood on Google is 50% of $WAOD$ value for Google. Hence, the collaboration strength or the influence of the Google on the committers in its direct neighborhood is twice as high as compared to the influence of all committers in Google's direct neighborhood on Google.

Figure 5 displays the weighted average in-degree and out-degree for the top 20 committers in external committers' network. Committers affiliated with the

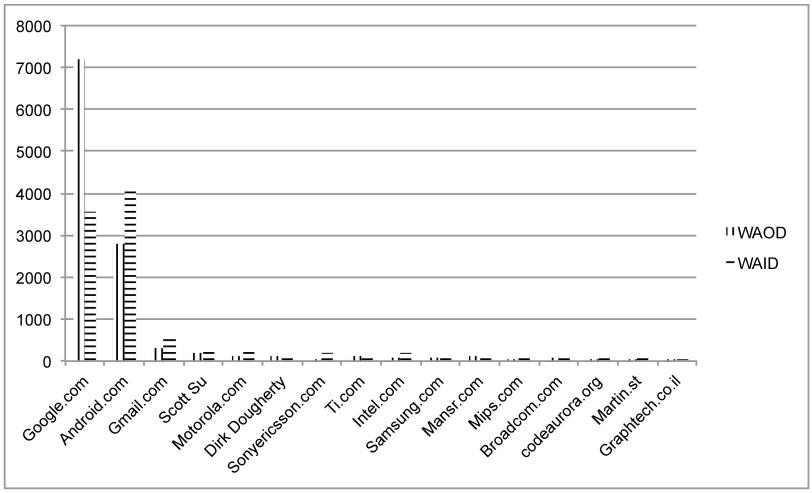

Figure 4: Weighted average in-degree and out-degree for top 16 committers in core committers' network

apple.com email address have the highest value for $WAOD$ metric. Some 30% lower value for the $WAOD$ metric have committers associated with gmail and google. The fourth and fifth highest value of $WAOD$ metric have committers affiliated with nondot.org an zuster.org. The highest $WAID$ value has Google, followed by committer associated with gmail.com email address.

Figure 6 shows the weighted average in-degree and out-degree for the top 20 committers in the combined, core and external committers' networks. Committers affiliated with the Google email address have highest value of $WAOD$. Some 30% lower value of $WAOD$ has Apple, followed by committers associated with gmail.com, nondot.org, and android.com.

The $WAOD$ and $WAID$ metrics indicate that for the entire Android source code base Google has the highest strength of co-affiliation with members in its direct neighborhood.

Figure 7 shows network centrality metrics values BC,CC, and EVC, for the core committers network. Committers with Google.com and Android.com have highest values for EVC, followed by committers associated with Gmail, Sony Ericsson, and Motorla. The value of BC is highest for committers associated with Google.com and Android.com, and decreasing sharply for committers associated with Sonyericsson.com and Motorola.com. Thus, the majority, i.e., 40% and 50% of shortest paths between two committers within the core committer network pass trough committers associated with Google.com and Android.com email addresses, respectively. The third and fourth highest values for BC metric have committers associated with Gmail.com and Sonyericsson.com email addresses, with the metric

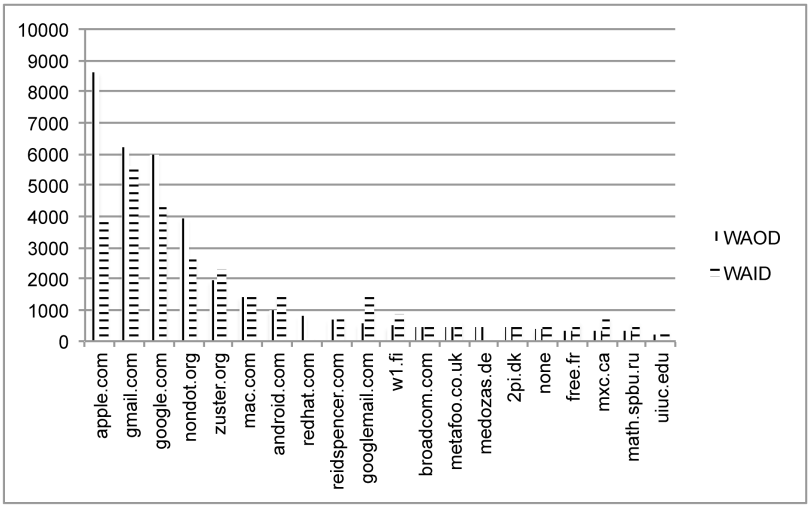

Figure 5: Weighted in and out degree for top 20 committers in external project committers' network

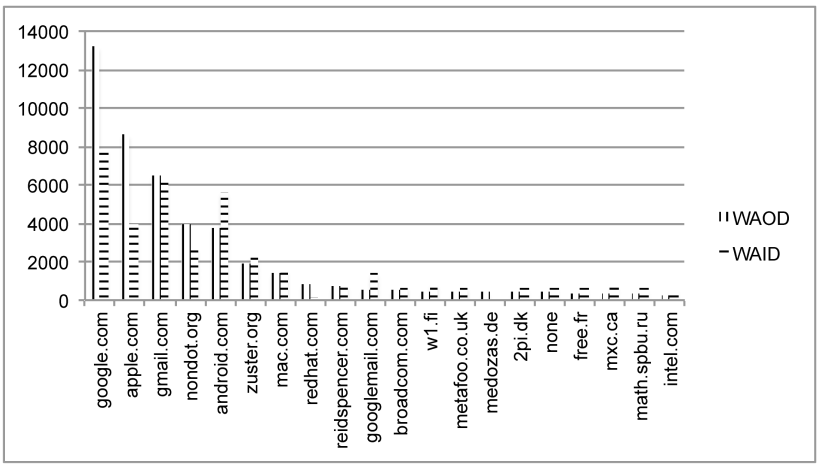

Figure 6: Weighted in and out degree for top 20 committers in core and external project committers' network

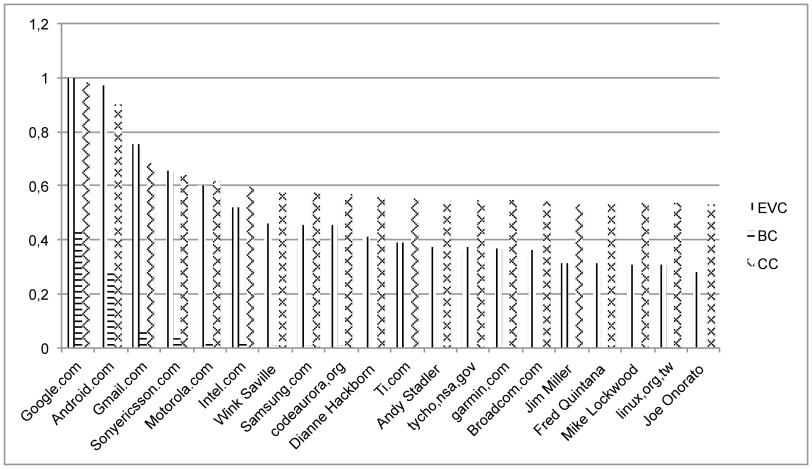

Figure 7: Network centrality measures for top 20 committers in the core committers' network

values indicating that only some 2% and 1% of shortest paths traverse through these committers, respectively.

Figure 8 shows network centrality metrics values BC,CC, and EVC, for the external committers network. The highest values for the EVC metric have committers associate with gmail.com, google.com, debian.org, non dot.org, apple.com, etc. Unlike EVC values for the core committers' network, the EVC values for external committers' network is more balanced and does not indicate as high of a differences among the top 30 committers. The BC value is highest for the committers associated with the google.com address, followed by the commuters associated with the gmail.com and debian.org address.

Figure 9 shows network centrality metrics values BC,CC,EVC for combined core and external components. Unlike EVC values for the core external committers network, the EVC values for the combined core and external committers' network show highest values for committers affiliated with google.com, followed by committers associated with gmail.com, intel.com, debian.org, codeaurora.org, etc address. The combined core and external committers' network shows more balanced values for CC and EVC, while the BC values indicated that some 40% of the shortest paths traverse through committers associated with a google.com address, followed by gmail.com with some 20% of the shortest paths, and intel.com with some 2% of the shortest paths.

Hence the metrics presented above in summery show:

Android core committers network The high average clustering coefficient and low graph density indicate that committers in a direct neighborhood of a committer are well linked. The identified potential sub-communities for the

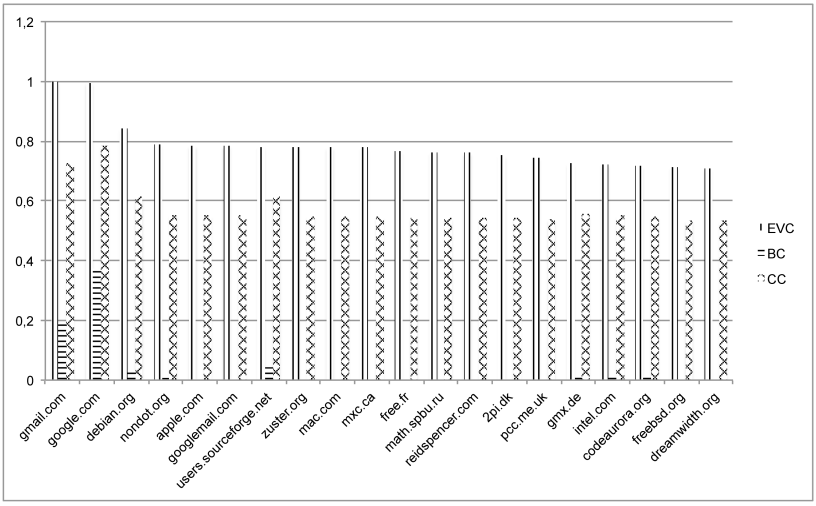

Figure 8: Network centrality measures for top 20 committers in external committers' network

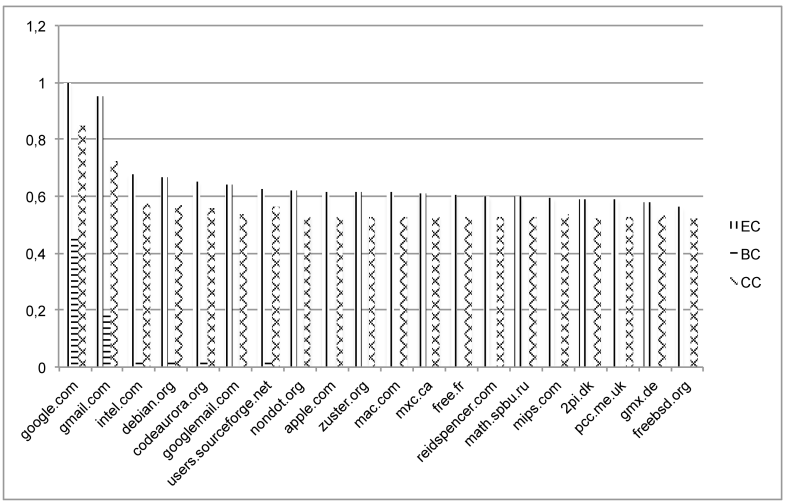

Figure 9: Network centrality measures for top 20 committers in combined core and external committers' network

core committer network should be disregarded since their existence is not statistically significant. The collaboration strength or the influence of the Google on the committers in its direct neighborhood is twice as high compared to the influence of all committers in Google's direct neighborhood on Google. The majority, i.e., 40% and 50% of shortest paths between two committers within the core committer network pass trough committers associated with Google.com and Android.com email addresses, respectively.

Android external committers network The number of sub-communities identified within the external committer network is 6 with the MC value of 0,356 indicating that existence of the 6 subnetworks is statistically significant. The EVC values for external committers' network is balanced among the top 30 committers. The BC value is highest for the committers associated with the google.com address, followed by the commuters associated with the gmail.com and debian.org address.

Android core and external network The number of identified sub communities for the combined, external and internal committers' networks is 7, with the MC value of 0,43 indicating that the existence of the sub-communities is statistically relevant. Committers affiliated with the Google email address have highest value of $WAOD$. Some 30% lower value of $WAOD$ has Apple, followed by committers associated with gmail.com, nondot.org, and android.com. Values for CC and EVC are balanced betweeing the top 20 committers, while the BC values indicated that some 40% of the shortest paths traverse through committers associated with a google.com address, followed by gmail.com with some 20% of the shortest paths, and intel.com with some 2% of the shortest paths.

Based on the results, the three committers networks show characteristics of highly centralized network structure, with committers affiliated with google.com, gmail.com, android.com, and apple.com being central to linking other committers. The four committers' affiliations also have the highest influence on other committers. Graph density metrics show that the core committers network and combined core and external committers network are not well connected with GD values of 0,058 and 0,055 respectively. The external committers network, composed of over 150 different OSPs has a twice as high value for the GD metric as for the Android OSPs core components network. As noted above, the highest value of GD metric is 1 indicating that all vertexes are connected.

The Figure 10, Figure 11, Figure 12 show graphical structures of core, external, combined core and external commiters' networks. The absence of subnetworks in the core committers network discussed above as well as low graph density can be seen in the Figure 10. The external network depicted in Figure 11 shows existence of subnetworks, which is expected for a source code base composed of different open source projects.

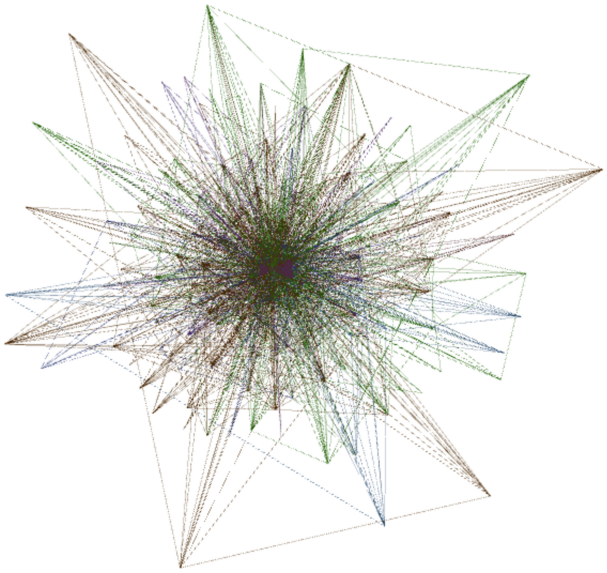

Figure 10: Core committers' network

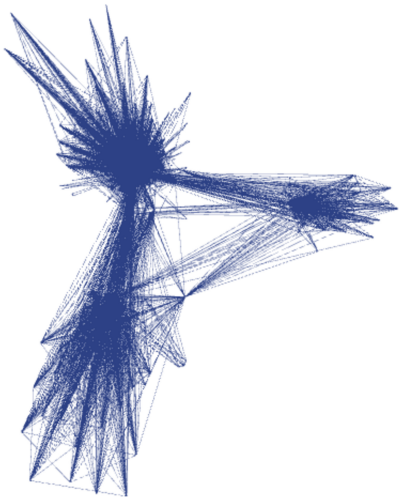

Figure 11: External committers' network

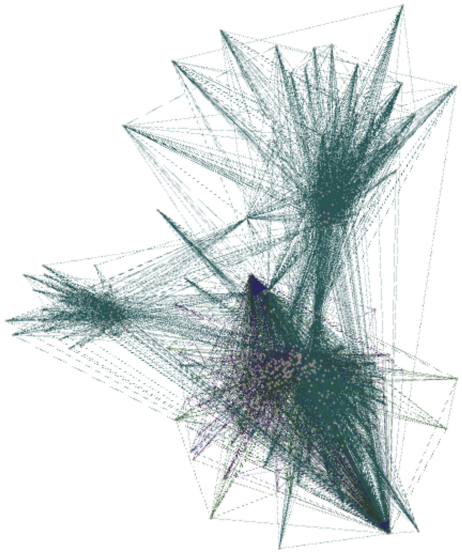

Figure 12: Combined core and external committers' network

3.2 Research question 2: What type of concerns should a company take into consideration when planning to become a contributor to the Android or a similar type OSP

Based on the results presented for research question 1, Android OSP exhibits characteristics of a highly centralized OSP, where committers with affiliations to google.com, gmail.com, android.com, and apple.com have the highest level of influence. The external open source projects that Android OSP hosts in its source code repository under the external top subdirectory shows committers affiliated with Google.com as being third most influential. The committers affiliated with Google have the highest BC value for the external committer's network, indicating that 40% of committers in the external committer network have shortest paths to other committers transversing through committers affiliated with Google. This is a compelling evidence that Google has been the most central, and the most influential in the Android OSP development not only for the core source components, but also for the external open source projects. The Android committers network has low graph density, i.e. low connectedness of committers, indicating low co-affiliation among committers.

From a perspective of a company that is planning to participate or participates in Android or a similar OSP this means that it should take into consideration that OSS product development tends to be highly influenced by one company. This

might indicate that the company planning to incorporate the Android into its product will need to work closely with Google to ensure that the changes it needs to see implemented in the source code base are included in a future OSS product releases. Google has built different sales models around the Android, primarily the GooglePlay store, the application market for Android devices, AdMob platform, and Web search. Hence, it is in the Google's interest to have the Android used and distributed on as many mobile devices as possible since this would mean higher revenues from its GooglePlay store, AdMob, and Search engine. However, the company should be aware that sales and marketing models change, and different alliances form. In order to influence and lead a large open source project, a company controlling the project development usually has a large development effort dedicated to the project. In case a company is no longer able to support the development is is possible that some other company takes the lead. Hence, it is possible for a company with highest control over the open source project to take the project in a direction not favored by some other project participants.

4 Discussion

The Android OSS project is an open source stack of software, i.e., a mix of OSS from over 150 OSS projects and components which Google initially purchased from Android corporation and later open sourced. Hence, different OSS projects have been used as a reusable software components to build a mobile device operating system used by companies from entire mobile ecosystem. While different open source solution stacks are not in itself a novel idea, e.g. [LAM], the Android OSS project stack is unique for several reasons. Firstly, it combines over 150 OSS projects of highly diverse nature into one integrated OSS product. Secondly, the integrated OSS product is of large scale, mostly maintained and developed by one company. Finally, this product has become the most used OSS product for mobile devices in 2012.

Based on the social network structure analysis results for Android committers's networks, it is evident that Google has the highest degree of influence and centrality on both, the core, and the external components. This shows how a large company with significant resources can create a large scale software products using other OSS components. In a company sponsored open source project the company invests a large development effort into the OSS product and there exists a possibility that the company might not be able to maintain the high level of development commitment. This possibility would also mean uncertainty for the future of the OSS product development, and, if realized, it can bring shifts in committers' influence on the project. This can create uncertainty on the future of the OSS product development, an important factor that should be considered by companies planning to join similar company sponsored projects. A company might decide to also closed source and license the open source product. Such situation can then

create a vendor lock-in effect, which contradicts a generally accepted notion of an OSS software product being free from vendor lock-in.

5 Conclusions

The conducted analysis have shown that Google has the major influence on the Android OSP. While it is favorable to use an OSS product as a commodity software, and thus decrease development costs by focusing available resources on developing differentiating parts of a product, at the same time this can raise many uncertainties. The future of OSS product whose development is highly sponsored and influenced by one company can come under the influence of market conditions the company finds itself in. This seems to go against the nature of OSS, which among other characteristics includes protection from vendor lock-in, i.e., high dependance of companies using Android on the Google.

More research is needed to understand and properly categorize OSPs in a way that would help the industry better understand own strategic position in a context of using an OSP to build business model. The research approach proposed in this study can be used as one way of studying a committer's network structure of a software development community.

Acknowledgment

This work was funded by the Industrial Excellence Center EASE - Embedded Applications Software Engineering, (http://ease.cs.lth.se)

References

[And05] AndroidCorp. *Google Buys Android for Its Mobile Arsenal*. `http://www.businessweek.com/stories/2005-08-16/google-buys-android-for-its-mobile-arsenal`. 2005.

[And07] AndroidOS. *Breaking: Google Announces Android and Open Handset Alliance*. `http://techcrunch.com/2007/11/05/breaking-google-announces-android-and-open-handset-alliance/`. 2007.

[Blo+08] Vincent D Blondel et al. "Fast unfolding of communities in large network". In: *Journal of Statistical Mechanics: Theory and Experiment* 10 (2008), P100.

[BH11] Stephen P. Borgatti and Daniel S. Halgin. "On Network Theory". In: *Organization Science* 22.5 (2011), pp. 1168–1181.

[Bra01] Ulrik Brandes. "A Faster Algorithm for Betweenness Centrality". In: *Journal of Mathematical Sociology* 25 (2001), pp. 163–177.

[Gep13] Gephi. *Open Source Software for Exploring and Manipulating Networks*. `https://gephi.org`. 2013.

[Han+M] Sudheendra Hangal et al. "All friends are not equal: Using weights in social graphs to improve search". In: *SNAKDD-2010: 4th SIGKDD Workshop on Social Network Mining and Analysis* (ACM, 2010).

[HOA11b] Martin Höst and Alma Oručević-Alagić. "A systematic review of research on open source software in commercial software product development". In: *Information & Software Technology* 53.6 (2011), pp. 616–624.

[HIC06b] James Howison, Keisuke Inoue, and Kevin Crowston. "Social dynamics of free and open source team communications". In: *Open Source Systems, IFIP Working Group 2.13 Foundation on Open Source Software*. 2006, pp. 319–330.

[Inc13] Google Inc. *Android Open Source Software Project*. `http://www.android.com/`. 2013.

[LAM] Open Source Community LAMP. *Linux, Apache, MySql, Python Open Stack*. `http://onlamp.com/`.

[LLC08] Frank van der Linden, Björn Lundell, and Gary J. Chastek. "Open Source Software Product Lines". In: *International Software Product Line Conference* (2008), p. 387.

[LLM09] Frank van der Linden, Björn Lundell, and Pentti Marttiin. "Commodification of Industrial Software: A Case for Open Source". In: *IEEE Software* 26.4 (2009), pp. 77–83.

[LF+06] Luis López-Fernández et al. "Applying Social Network Analysis Techniques to Community-Driven Libre Software Projects". In: *International Journal of Information Technology and Web Engineering* 1.3 (2006), pp. 27–48.

[Ray01a] Eric S. Raymond. *The Cathedral and the Baazar*. O'Reilly Media, Inc., 2001.

[RH08] Per Runeson and Martin Höst. "Guidelines for conducting and reporting case study research in software engineering". In: *Empirical Software Engineering* 14 (2008), pp. 131–164.

[Toi+07] Riitta Toivonen et al. "The role of edge weights in social networks: modeling structure and dynamics". In: *Proc. International Society for Optics and Photonics, SPIE* 6601 (2007).

[WS98] Duncan J. Watts and Steven H. Strogatz. "Collective dynamics of 'small-world' networks". In: *Nature* 393.6684 (1998), pp. 440–442.